HZ BOOKS
华章教育

计 算 机 科 学 丛 书

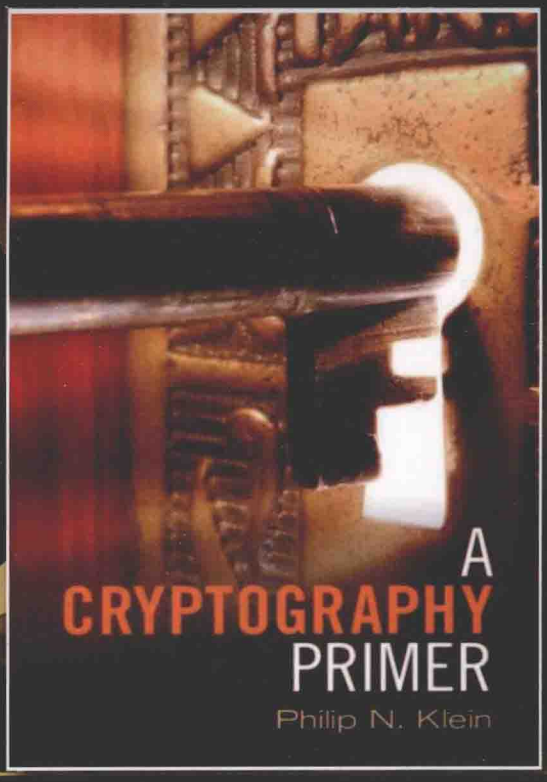
CAMBRIDGE

密码学基础教程

秘密与承诺

[美] 菲利普 N. 克莱因（Philip N. Klein） 著
徐秋亮 蒋瀚 王皓 译

A Cryptography Primer
Secrets and Promises



机械工业出版社
China Machine Press

密码学基础教程 秘密与承诺

A Cryptography Primer Secrets and Promises

密码学在尤里乌斯·凯撒时代就被用于军事和外交领域。在互联网时代，密码学最广泛的应用领域也许是商业——从保护电子传输的安全性到保证通信免受工业间谍的威胁。

本书是一本针对本科生的导论教材，非常容易阅读和理解，解释了用来实现通信隐私性的密码协议，以及如何使用保证消息、文档或者程序的正确性、完整性及来源可靠性的数字签名。作者提供了一个关于现代密码学基本原理和数学知识的导引，让读者看到密码技术的本质和被认为安全的原因，而不是提供关于如何配置Web浏览器及电子邮件程序的基本知识。

作者简介

菲利普 N. 克莱因（Philip N. Klein） 布朗大学计算机科学系教授，曾获得美国国家自然科学基金会青年科学家总统奖，以及多项美国国家自然科学基金资助。他因为图算法研究中的贡献成为ACM Fellow，并因为在科学教学上的优异表现获得布朗大学Philip J. Bray奖。

CAMBRIDGE
UNIVERSITY PRESS
www.cambridge.org

投稿热线：(010) 88379604
客服热线：(010) 88378991 88361066
购书热线：(010) 68326294 88379649 68995259

华章网站：www.hzbook.com
网上购书：www.china-pub.com
数字阅读：www.hzmedia.com.cn

2016

上架指导：计算机/信息安全/密码学

ISBN 978-7-111-54436-4

9 787111 544364 >

定价：49.00元

计 算 书

密码学基础教程

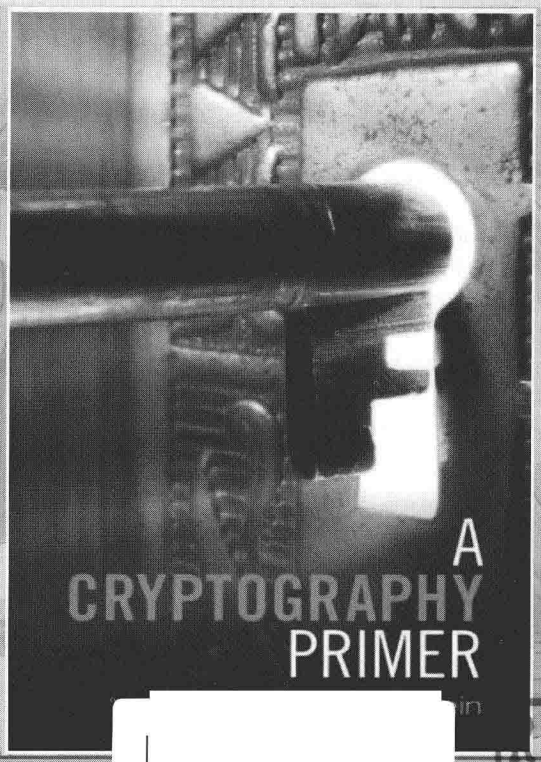
秘密与承诺

[美] 菲利普 N. 克莱因（Philip N. Klein） 著

徐秋亮 蒋瀚 王皓 译

A Cryptography Primer

Secrets and Promises



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

密码学基础教程：秘密与承诺 /（美）菲利普 N. 克莱因（Philip N. Klein）著；徐秋亮，蒋瀚，王皓译 . —北京：机械工业出版社，2016.9
（计算机科学丛书）
书名原文：A Cryptography Primer：Secrets and Promises

ISBN 978-7-111-54436-4

I. 密… II. ① 菲… ② 徐… ③ 蒋… ④ 王… III. 密码－高等学校－教材 IV. TN918.1

中国版本图书馆 CIP 数据核字（2016）第 176191 号

本书版权登记号：图字：01-2016-3495

本书以通俗、直观的方式清晰阐述了现代密码学的基础知识，包括用来实现通信隐私性 / 机密性的保密算法（协议）及保证消息正确性、完整性、来源可靠性的数字签名协议，提供了一个易读、易学的关于现代密码学基本原理和数学知识的导引，通过浅显的例子和生动的语言让读者绕过晦涩的专业术语而直接看到密码技术的本质。

本书叙述清晰，简单易懂，适合作为高等院校计算机及相关专业本科生教材。

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：迟振春　　责任校对：董纪丽
印　　刷：北京诚信伟业印刷有限公司　　版　　次：2016 年 9 月第 1 版第 1 次印刷
开　　本：185mm × 260mm　1/16　　印　　张：10.75
书　　号：ISBN 978-7-111-54436-4　　定　　价：49.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

文艺复兴以来，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的优势，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭示了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短的现状下，美国等发达国家在其计算机科学发展的几十年间积淀和发展的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起到积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章公司较早意识到“出版要为教育服务”。自 1998 年开始，我们就将工作重点放在了遴选、移译国外优秀教材上。经过多年的不懈努力，我们与 Pearson，McGraw-Hill，Elsevier，MIT，John Wiley & Sons，Cengage 等世界著名出版公司建立了良好的合作关系，从他们现有的数百种教材中甄选出 Andrew S. Tanenbaum，Bjarne Stroustrup，Brian W. Kernighan，Dennis Ritchie，Jim Gray，Afred V. Aho，John E. Hopcroft，Jeffrey D. Ullman，Abraham Silberschatz，William Stallings，Donald E. Knuth，John L. Hennessy，Larry L. Peterson 等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及珍藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力相助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专门为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近两百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍。其影印版“经典原版书库”作为姊妹篇也被越来越多实施双语教学的学校所采用。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因

素使我们的图书有了质量的保证。随着计算机科学与技术专业学科建设的不断完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都将步入一个新的阶段，我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

华章网站：www.hzbook.com
电子邮件：hzjsj@hzbook.com
联系电话：（010）88379604
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

华章科技图书出版中心

计算机信息网络已经成为人们社会生活的基础设施，人们对网络的依赖已经可以和水、电等基本生活必需品比肩，大量的生活信息、工作信息和社会信息在网络中高速传送、处理和应用，多维度地支撑着人们家庭及社会生活的方方面面，形成了人们生活的另一个“空间”，财产、身份、学历甚至整个社会关系都变成了信息网络空间中的一条条“记录”。但是，人们对信息网络的强烈依赖，也产生了另一方面的问题。存储于网络空间中的数据便于传输、处理、共享但可控性差。网络空间给人们带来便利的同时也带来了隐私信息泄露、商业信息泄密以及消息被篡改、不当使用、身份被假冒等多重风险，如果不能对信息进行很好的保护，将会造成不可估量的损失。

密码学是一门古老的学科，甚至可追溯到古埃及的法老时代。当然，最初的密码应用环境非常简单，是一种主要用于保证通信安全的技术或技巧，直到20世纪40年代香农创立信息论并以此研究保密系统开始，密码的设计和分析开始变为一种有理论基础、有科学方法的“科学”。而目前信息网络应用的普遍性和其中信息安全问题的广泛性，使得密码学不再是军事、外交、国防领域的专宠，而成为人们社会生活中时时处处依赖的信息安全保障必备工具，这也是从20世纪70年代以来密码学迅速发展的原因。密码学已经成为当今社会不可或缺的应用学科。

为了使网络使用者对密码学有一个初步了解，我们翻译了美国布朗大学 Philip N. Klein 教授编著的《A Cryptography Primer: Secrets and Promises》一书。与普通的密码学教程不同，这本书不以密码学的研究和应用为目的，而注重密码知识的普及。阅读该书几乎不需要任何专门知识，书中从最初步的概念、最易理解的例子、最简单的应用开始，深入浅出，层层递进，一直到密码学思想的实质。本书不追求表达的严谨性和方法的通用性，力求让读者理解密码学最本质的原理，是一本极具特色的入门书，通俗易懂而又极其深刻。译者在翻译该书的过程中深感获益，希望该书中文版的出版能够给希望了解密码学的中文读者带来帮助。

对该书的编著者 Philip N. Klein 表示敬意。

译　者

2016年7月27日

前　言

A Cryptography Primer: Secrets and Promises

作为一个数论专家与和平主义者，G. H. Hardy 在其自传《一个数学家的致歉》中写道：

> ……令高斯以及少数数学家们欣慰的是，至少还有一种科学“数论”……能够远离人们的日常活动，它应当保持纯粹和优雅。

Hardy 的这本书于 1940 年出版，他当时正面临着职业生涯的结束。如果他能够延后 30 年再做出论断的话，或许他会得出截然不同的结论，因为数论成为一项与战争相关的重要技术（密码学——研究秘密编码的应用学科）的基础。

密码学的应用至少有数千年的历史。在印度圣经中，密码学被列为 64 项女人的技艺之一。其中一个著名的初等密码系统要归功于尤里乌斯·凯撒。许多逸闻趣事证实了多年以来密码学和密码分析学（即编码破解）在战争和外交博弈中的重要作用。例如，英国人曾截获并破译了由德国外交部发给墨西哥政府（经由大使）的齐默曼电报，在电报中德国许诺将得克萨斯州、新墨西哥州以及亚利桑那州划归给墨西哥作为其帮助对抗美国的回报，这促进了美国加入第一次世界大战的进程。密码分析学同时也在不那么重大的事件中发挥着作用，以下文字摘自 Casanova（1757）的自传：

> 五六周之后，她问我是否已经解密了这些手稿……我告诉她是的。
>
> “先生，恕我冒昧。没有密钥，这怎么可能呢?”
>
> “希望我告诉你密钥吗，夫人?”“如果这样的话告诉我吧。”
>
> 我告诉她密钥，这个密钥并不属于任何语言，然后看到她吃惊的表情。她告诉我说这是不可能的，因为她坚信她自己是唯一掌握这些密钥的人，她仅仅将这些密钥记在脑海中，从来没有写下来。
>
> 我本可以告诉她真相的——已被我掌握的与解密手稿同样的计算已经让我知道了密钥——但是一个邪恶的想法闪过心头，阻止我告诉她这一真相。我保持神秘使她深深地被我俘获，那天我主导了她的灵魂，肆意地散发着我的力量。

然而在信息时代，或许密码学最大的贡献还是在商业领域。一直以来，银行用密码学来保障电子传输的安全，分布在不同地域的公司用密码学来保证他们之间的通信不被工业谍报人员窃听。但是，或许最令人兴奋的应用在于，让素未谋

面并因此无法提前协商密钥的双方也能够安全地通信。随着互联网上的商业活动日渐繁荣，这种应用变得更加普及。幸运的是，现今的指数密钥交换协议和公钥密码学能够使这类应用成为现实。

Diffie 和 Hellman 在 1976 年提出了公钥密码学，其思想在于：有两个不同密钥，公开密钥用来加密消息，而私有密钥则用于解密；由一方秘密地生成这对密钥，他可以将加密密钥公开而不暴露用于解密的密钥。因而任何人都可以给密钥生成者发送加密的消息，而只有密钥生成者能够解密这一消息。Rivest、Shamir 和 Adleman 于 1978 年最先实现了这一思想。至于他们的方案有多么闻名于世，我们来看看下面这段摘自滑稽喜剧《山河之旅》中的对白吧：

> “杰伊，我不太熟悉计算机，对此知之甚少。我了解到这个密码是两个差不多 100 位的大素数相乘得到的，对吧?”
>
> “是的，很对。这被称为 RSA 密码系统。”
>
> “好吧，这一名字来源于 MIT 的 Rivest、Shamir 和 Adleman。我只知道这么多。我也知道即使是使用先进的计算机来破解，也需要花费无穷无尽的时间，”她回忆着，“两个 100 位的素数相乘得到的密钥差不多需要 38 亿年的时间来破解，对吧?”“完全正确。很明显，所有被窃取的信息都来自于从公司办公室到你家的电话线路上发生的窃听。假设只有麦克掌握解密密钥，如果他不将这一密钥分发给别人，那么是没人能够解密这一编码的。但是这一说法在逻辑上还有一点不严谨之处，”他边说边松了松深绿色的丝织领带，“Vee，这里比我想象的热好多啊，你介意我脱掉外套吗?”
>
> “当然不，你太客气了。”她说道……

我们的女主角 Vee 说道，RSA 的安全性是基于分解两个素数乘积的困难性，因此，它使得 Hardy 最喜欢的“纯”数学领域（数论）有了用武之地。这一密码系统（同大多数密码系统一样）的根基在于简单与困难的区别。生成一个公钥/私钥对就如同选择两个 100 位的素数并将它们相乘那么容易，而正如 Vee 所言，攻破这一系统（利用当前已知的方法）则需要大量的时间才能完成；这似乎需要用两个素数的乘积来确定这两个素数分别是什么，这一问题称为整数分解。尽管针对这一问题的研究有着持续的进展，但是无论如何，已知的算法的速度都还没有快到足以威胁 RSA 安全性的程度。下面引用一个更了解市场营销手段而不精通数论的人的一句话：

> 由于数字货币系统的隐私性和安全性都依赖于密码学，因此任何能

> 够攻破密码系统的数学或计算机科学的突破都会是一场灾难。而在数学方面最为显而易见的突破或许就是构建一个能够快速分解大整（素）数的方法。——比尔·盖茨，《未来之路》第一版，265 页
>
> （大整数分解是这样的问题，即已知一个数，求这个数是由哪些素数相乘得到的，如果这个数是一个素数，那么分解的结果就是它本身。）

但是 RSA 不仅仅用于加密。正如 Diffie 和 Hellman 意识到的那样，公钥密码学的另一个方面在于数字签名。使用类似于 RSA 的方式，公-私密钥对的生成者可以用私钥对文档产生一个签名。这是一个由文档产生的数字，拥有公钥的人都可以验证这一签名与文档是一致的（满足某种数学关系），而且只有拥有私钥的人才能够为文档产生合法的签名。因此一个文档的合法签名可以作为密钥生成者需要为此负责这一事实的强有力的证据。如果有人篡改了文档，那么篡改后的文档和签名不会再有相同的数学关系了，因此这一文档将会被视作无效。于是，数字签名可以用于互联网中传输的消息的认证，以此来防止潜在的消息篡改和伪造。它们可以用于创建不可伪造的证书，如电子版的信用卡或护照。它们也可以用于检测对计算机程序未经授权的篡改，如病毒的侵入等。

与此同时，其他保障计算机安全的技术也被陆续提出，包括对身份的安全认证（这与在电话中询问某电话号码、信用卡号码等信息以确认对方身份类似），对一个文档内容承诺但不泄露其信息的方法（对一个密封的信封的模拟），为一个文档打上时间标记的方法（对给自己邮寄一封信以得到带有时间的邮戳这一方法的模拟）。

计算科学对计算简单问题与计算困难问题进行了分类，而密码学技术正是构建于计算科学这一理论之上：在你拥有密钥的情况下，解密编码是容易的；而如果你没有密钥则不然。密码学正是对这种智慧追求的具体实现。

为了让更为广大的读者都能接触到这一饶有趣味的、令人兴奋的而且越来越重要的充满智力挑战性的领域，我开设了课程“秘密与承诺：数字安全导论”。我为这门课撰写了这本书作为教材。题目中“秘密”这个词表示用密码学的方法来实现私密通信；“承诺”这个词表示用数字签名来保证消息、文档或程序的有效性与完整性。本书旨在对现代密码学的基本理论和其所依赖的数学基础给出一个简介，而并非一部面向实践的、教你如何做的教材，即这本书不会指导读者如何用现代计算机程序（如网页浏览器和电子邮件系统）来实现数字安全。这些程序一直在更新换代，而且如果想在市场上占有一席之地，必须做到让它们的使用者无须了解这些软件所依赖的安全技术便可轻松使用。在本书中，我们要透过现象看到问题的本质，研究安全技术，探寻其之所以安全的原因。

对于一些基本的密码学方案，如 DES 和 SHA，其细节并没有启发作用，因而在本书中我们省略对其细节的讨论。其他一些基于初等数论的方案也能发挥与之相同的功能，尽管这些基于数论的方案效率低下难以实用，但是它们被认为是安全的，而且适合在这门课中讲解。因此为了最大限度地做到可读性与一致性，我们在实用性方面做出一定的牺牲与权衡，对于急切想了解 DES 细节的读者可以在其他教材中轻易地找到这些知识。

致谢

非常感谢 Sarah Finney、Peter Galea、Kevin Ingersoll 和 Mark Weaver 帮助我完成基于该书的课程。同样感谢国家科学基金对该课程建设的资助，感谢 Michael Yanagisawa 帮助书稿的校对，最后还要感谢 Alice、Bob、Eve 以及其他频繁出现在现代密码学文献中的角色。

目 录

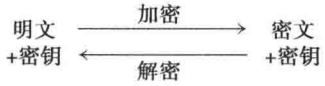
A Cryptography Primer: Secrets and Promises

第1章
A Cryptography Primer: Secrets and Promises

引　论

1.1　加密与解密

我们最熟悉的密码学用途是隐藏一个消息或文档的内容，*密码系统*（cryptosystem）就是能够实现这一功能的系统。它包括两个部分：加密方法和解密方法。未经变换的、可读的消息或文档称为*明文*，经过变换的、一般不可读的形式称为*密文*（密码系统“cryptosystem”在英文中也被称为“cypher”，或者拼写为“cipher”）。



加密（encryption）是将明文转化为密文的过程。加密方法需要两个输入：明文和密钥。类似地，解密方法也需要两个输入：密文和密钥，并输出明文。

在传统的密码系统中，加密和解密使用相同的密钥，这样的系统被称为*对称密钥密码系统*（symmetric-key cryptosystem）。如果 Alice 想要发送一条加密消息给 Bob，双方都必须知道所使用的密钥。（这一点与*公钥密码系统*不同。在一个公钥密码系统中，有两个不同的密钥，一个用于加密，另一个用于解密。我们将在后面讨论公钥密码系统。）如果 Alice 和 Bob 希望他们的通信内容是保密的，就必须保证密钥的秘密性，因为对于任意窃听者而言，如果他知道密钥并截取了密文，就可以确定明文消息。

我们在这里假设每个潜在的窃听者都知道加密和解密的方法，在本书 1.4 节中将要详细讨论这个假设。这个假设在方法论上是现代密码学的基础，我们将这条假设贯穿于全书。简单地讲，*我们总是假设每个潜在的攻击者了解你所使用密码系统的每个细节*。

密码分析（cryptanalysis）是试图攻破密码系统的过程，窃听者采用密码分析的方法来发掘 Alice 发送给 Bob 消息的内容。在 1.6 节中，我们简述窃听者可

能采取的几种不同类型的攻击方法，但是密码分析的细节知识超出了本书讨论的范围。

1.2 信道、安全与不安全

我们常常使用各种不同的通信媒介，如电话网络、无线电波、有线电视网络、计算机局域网、互联网和印刷媒体等。银行通过一个网络将他们的自动取款机连接到中心计算机和另一个网络以实现电子资金转账；寻呼服务使用有线和无线通信的结合；而某些卫星和地球之间的通信则使用微波。我时常使用计算机内存在现在的我和将来的我之间通信；当我使用电话的时候，在我的声音到达听筒之前，声音穿过了空气；而在拨号的行为中，我是与电话系统进行通信。

我们希望将数字安全的概念应用于所有这些通信媒介中，为了达成这一目的，我们从纷繁复杂的差异中抽象出一个单一的、通用的术语——信道。信道是通信双方之间的媒介（我喜欢想象成连接两个铁罐的绳）。

当然，大多数通信媒介允许两方以上的实体间相互通信，然而在大多数情况下，认为信道连接两方在概念上就足够了，因为如果有多方参与的话，我们可以认为存在多条信道。

一个人将一个特定通信媒介（比如电话网络）看作是安全的或者是不安全的是根据他自己的观点而定的。比如说，在通常情况下，我们认为电话网络是相当安全的，然而，每年都有数以百计的政府窃听和监听命令被批准，而每一个命令都会导致平均数千段通话处于被窃听之中。

出于学习密码学的目的，我们将简单地宣称一个信道是安全的或是不安全的。如果存在第三方（窃听者）可以截取（窃听）信道中传输的消息，我们认为该信道是不安全的。在某些情况下，窃听者甚至可能篡改从发送方到接收方的消息。

安全信道是指不会受到窃听与干预的信道。当然，人们更感兴趣的是将密码学应用于不安全信道。幸运的是（或者不幸的是，取决于你怎么看），在我们的现实生活中不安全信道比比皆是。在本小节剩余部分，我们将对三种通信媒介描述某些导致其不安全的特征。这些只是作为例子，毋庸置疑，读者也可以在其他通信媒介中找到不安全的因素，包括本节开头提到的各种通信媒介。

1.2.1　互联网

当今密码学最显而易见的应用都涉及互联网，原因如下。首先，互联网在某种程度上是由计算机组成的，而计算机擅长处理密码学相关任务。其次，互联网是使先前未曾熟识的各方实现自主通信的完美媒介。最后，也是最重要的，互联网的特殊结构使得它需要一套安全防护机制。在互联网中，计算机极少与其他计算机直接相连，当你使用位于罗德岛卧室中的计算机发送一条消息到父母位于加利福尼亚的计算机时，你的这条消息将会经过很多中间计算机，每台中间计算机都会尽力将消息向前转发给更为靠近目的地的另一台计算机，同时存在一套机制检测这条消息是否最终找到正确路线。中间计算机在转发这条消息之前，可能存储这条消息的副本，或者将其篡改后再转发。然而，在系统中没有任何措施阻止它们这么做。一台流氓计算机甚至并不转发这条消息，却返回一条报文显示你的原始消息已经完成了传递。（当然，情况并不像这种描述那样可怕，因为报文的传递路径经常是不可预测的，你的消息通常被分成许多片，每一片会沿着不同的路径被转发，而且，大多数路径上的中间节点计算机可能并非恶意。）

一个更为严峻的情形出现在远程登录中。如果你在加利福尼亚的一台计算机上有一个账户，通过一个叫作“远程登录”（telnet）的程序，你可以在罗德岛的电脑登录这台在加利福尼亚的计算机。当然，加利福尼亚的计算机将向你发送一条消息，要求你输入登录口令。当你提供了口令之后，这条口令会穿过罗德岛与加利福尼亚之间的所有中间计算机。任何一台中间计算机都可以存储你在加利福尼亚这台计算机的地址、你的用户名和你的口令，随后，他们就获得了登录你账户的权限。有证据表明，有数万条用户口令被流氓计算机用这种方式所截获。

由于互联网被广泛用于商业目的，因此面临的威胁日益增加。假设你在做出一个商业决定之前，用网页浏览器获取最新的股票信息，有可能（尽管不都是这样）你的商业对手已经建立了一台计算机来代替他真实的设备，伪造了你正在搜集的股票数据。假设你要下载喜欢的电脑游戏公司的演示程序，有可能一台流氓计算机会截取浏览器发出的请求，并发送给你一个经过篡改的、感染了病毒的版本。最后，假设你正在浏览一个在线书店的网页，网站提供了一个方法以加密你的信用卡号，而你也将信用卡号提供给了这个网站。但是事实

上，你所浏览的网页可能根本就不是这个在线书店，而是在一台流氓计算机上伪造的页面。

1.2.2 局域网

不仅仅像从加利福尼亚到罗德岛这样的远距离通信存在安全隐患，连接你的电脑与服务器（用于存储程序以及提供邮件服务等）的局域网也可能是不安全的，接入网络的流氓计算机可能会侵入你的通信（使用一种被称之为“数据包嗅探器”（packet-sniffer）的软件），甚至注入经过篡改的数据。

1.2.3 移动电话

当然，移动电话利用空中电波进行通信，因而可能被窃听，如美国众议院议长纽特·金里奇在1997年1月发现其电话遭受窃听。不仅通话会遭受窃听威胁，当一个呼叫被发起时，移动电话会传输一个账号用于收费，这同样是危险的。1995年，移动电信业遭受了约四亿五千万美元的诈骗损失[⊖]，其中大部分归咎于手机“克隆”，即犯罪分子截获手机账号，并将这个账号植入另一个手机中，因此便“克隆”了原来的手机。用新的手机打电话，将会被计费到原来的手机。

移动电信业界正在采取措施，通过加入安全特性来防止诈骗。然而，旧的通信标准难以废弃，更进一步地，正如1.3节中将要讨论的那样，引入的安全机制也没有被充分证明。

即使一个用户并不在使用手机，安全威胁依然存在。当一个手机漫游的时候，它将会向系统注册位置变动情况，这些信息的传输可以被截获，并被用来帮助确定这个手机（从而也是其机主）的位置。联邦调查局要求建立标准，强制要求手机能够为警方提供机主的位置信息[⊜]。

1.3 隐匿式安全

考虑以下场景：

- 在第二次世界大战期间，美国军方在太平洋战区雇佣印第安纳瓦霍人来

⊖ 来自：贝尔大西洋公司，http://www.ba.com/nr/95/may/freddie.html。
⊜ 执法中的通信技术支持，参见 http://www.epic.org/ privacy/wiretap。

从事安全通信的任务。没有采用密码学技术来确保通信中的机密性，采用纳瓦语进行通信被认为是足够安全的。就目前所知，该方案的安全性从未被攻破。

- 在埃德加·艾伦·坡的小说《失窃的信》中写道，同名的信被藏在一个很“显眼”的地方——对警察来讲是被隐藏起来的（但是对本故事的主人公而言，是不被隐藏的）。
- 我的一位在另一所大学工作的富有想象力的同事希望将期中考试试卷的初稿分享给助教，以便做出修订，因而他将期中考试试卷放在公共文件夹，但是将这个文件夹命名为“systemstats”，认为不会有人不嫌麻烦地来翻看这个文件夹。

这些都是“隐匿式安全”的例子，其思想在于通过保持特定机制的秘密性来获取安全性。这一概念（虽然不一定都采用这一名称）具有悠久的历史——一个充斥着无数失败的历史。当一方（通常为政府机构）依靠一种秘密的方法获得安全性时，另一方通常能够通过刺探情报、碰运气或是长时间的工作来探知他们用了何种方法。发生在20世纪的例子有英国和波兰成功地攻破德国的恩尼格玛（enigma）密码系统，以及美国成功地攻破日本的“紫色”密码机（07-shiki O-bun In-ji-ki）[㊀]。

安全方案的开发者依然在寻求通过隐匿来获取安全性。追溯到1997年3月在数字移动电话系统中使用一种密码系统来加密用户通过按键输入的数字（比如说拨号）。最初本以为这个密码系统的细节仅有业内工程师知晓，然而这些技术细节却被泄露并公布在网上。随后有研究者指出，该系统远不及预期的那样安全[㊁]。

通过隐匿的手段来获取安全性是个错误，尤其是在数字化信息安全无处不在的时代。当数字化系统广泛用于商业中时，大多数安全通信发生在从未接触过的双方之间。在商业行为中，只有被广泛分发的安全机制才能是有用的。在这种情况下，如果假设这种安全策略的细节不会落入有能力发现和利用安全漏洞的人之手，这也未免太过于乐观了。

㊀ Kahn，《破译者》（The Codebreakers）。
㊁ http://www.counterpane.com/cmea.html。

1.4 另一种选择：柯克霍夫原则

考虑到隐匿并非获取安全性的可靠途径，那么我们有什么替代的方法吗？毕竟有些东西是必须保密的。

这个问题的答案（或者至少是这个问题的部分答案）在1881年由一个了不起的学者给出了清晰的阐述，此人当时是巴黎大学的德语教授[⊖]。他在出生后起名为简·纪尧姆·休伯特·维克多·弗朗索瓦·亚历山大·奥古斯特·柯克霍夫·冯·纽汶霍夫（Jean-Guillaume-Hubert-Victor-François-Alexandre-Auguste Kerckhoffs von Nieuwenhof），但是随后他将名字缩短为奥古斯特·柯克霍夫。他的职业生涯大部分在教英语和德语，偶尔也教意大利语、拉丁语、希腊语、历史学和数学。他出版的书包括语法学、德国戏剧的起源、艺术与宗教的关系，当然也包括密码学。在《军事密码学》一书中，他意识到之前提出的密码系统远远不够安全：

……我非常震惊地看到，我们的专家和教授依然在教授和推荐在战时使用的一些密码系统，而这些密码系统即使是最没有经验的密码分析学家都能够在不到一小时内找到其密钥。

柯克霍夫意识到，关于安全性的欺骗性论断通常基于如下误导性的推断：系统地尝试所有的密钥需要几个世纪的时间，这一观点促使了对密码学的忽视与轻信：

6

……或许……可以认为，某些作者的信口断言比完全缺少关于秘密书写技术的严肃著作更大地加深了对于密码系统价值的错误观念。

从而柯克霍夫认识到，一个密码系统的安全性通常并不应该由它的开发者经过纯粹的推理而建立，而是应该经受独立的密码分析者严酷的攻击测试来获得[⊜]。他提出现在我们称之为柯克霍夫原则的一条准则：密码系统的安全性应该仅仅取决于所使用的密钥的保密性，而不是对该方案本身的保密。

柯克霍夫原则要求假设每一个潜在的攻击者和黑客都能够了解你所使用的密码系统的所有细节。这是个悲观的假设，但它确确实实是需要的。正如

⊖ 此材料来自于《破译者》。

⊜ 在探讨完美安全时，我们会看到一个例外。

Schreier 所说，“人们可以假设中央情报局不会习惯于告诉摩萨德（以色列的情报机构）他们所使用的密码算法，但是摩萨德可能通过任何方法来获取这一算法。”柯克霍夫原则尤其适用于被大量人群使用的安全系统（比如网页浏览器中的安全系统），因为所有这些人必须被授予至少是隐式访问系统细节的权限——而你的敌人，或许就在这些人之中。

柯克霍夫原则是诱人的，但是如果没有一个密码系统能够满足该原则的严格要求的话，这一原则依然没有任何意义。对于构造安全性如此之高的密码系统，以至于对于完全了解该系统的敌手都无法攻破，我们应当抱有多高的期望呢？正如埃德加·艾伦·坡在著名的小说《金甲虫》中描述的那样，“或许有理由怀疑，人类的智慧是否能够创造人类无法通过恰当方法解决的谜团。”就像我在本书中希望要表述的，我们有理由相信安全的密码系统事实上是确实存在的。

1.5　密码学分类

尽管从狭义上来讲，密码学是研究秘密书写（即加密）的学科，但是它已发展成为数字化安全领域中囊括了多项技术的一个通用的术语。在密码学研究中，在不同的概念层次之间进行区分是有帮助的。目前，将其分为如下五个类型就足够了：

1. 含糊的安全目标
2. 形式化的安全目标
3. 协议
4. 密码学构建模块
5. 密码学构建模块的实现

*含糊的安全目标*是密码学最抽象的元素，也促使其他几个层面的出现与发展。“我希望和他人的通信是私密的。”“我需要一个不可伪造的文件。”“我希望确信对计算机程序的任何修改都是可检测的。”虽然密码学的方法能够实现上述目标，但是这些说法太过笼统，我们无法验证这些目标是否已经达到。比如，为了保证通信的隐私性，我们必须确保任何窃听者都不能获知通信的内容。对于一个潜在的窃听者来说，哪些信息和资源是可用的？他能够采取什么样的行为？窃听者是否能获取通信的内容严重依赖于上述问题的答案。

为了语言描述更为精确以及更加专用，我们必须致力于*形式化的安全目标*。

一个形式化的安全目标表明了一个能够采取特定攻击手段、拥有特定资源（比如时间、内存、计算能力）的攻击者无法攻破该系统。我们将在 1.6 节中讨论各种攻击类型。学习了概率论和计算复杂性的相关知识之后，我们可以深入了解形式化安全目标。

密码协议是指多方之间进行通信的一系列规则，以实现某些密码学目标。例如，接下来我们将要讨论到双方如何通信以安全地协商一个密钥；我们将实现这一功能的协议称为*指数密钥协商*（exponential key agreement）。人们可以将“加密-发送-接收-解密”这一过程看作一个非常简单的协议：一方将明文加密，随后发送给另一方，另一方将其解密。另一个用于认证的简单协议是计算机登录：一个用户发送消息要求访问计算机；计算机回应要求用户输入口令；这个用户发回口令；计算机发回一个消息授权用户访问。我们同样也将学到复杂的认证协议。

协议由*密码学构建模块*构成。我们最为熟悉的密码学构建模块便是密码系统，但在本书中，我们同样也将会讨论其他类型的密码学构建模块，包括单向函数、消息摘要函数和数字签名系统等。人们通常通过非形式化的或者形式化的安全目标将每个密码学构建模块联系在一起。

区分概念上的构建模块和这些模块的特定实现经常是有帮助的。举例来说，
8 有很多不同的密码系统，但是它们在实现安全目标上起着相同的作用。现代密码学的部分力量正来自于密码学研究者在抽象的层面上区分和描述了有用的密码学构建模块。

抽象使得密码协议的开发者能够实现更大的普适性。例如，当协议需要用到一个密码系统时，开发者无须指明具体用的是哪个密码系统（无须说“利用密码系统 DES 进行加密……”），而仅仅说明所用到的密码系统满足何种安全目标即可。另一方面，这种普适性使得协议更为健壮：如果有一天 DES 的使用被废止了（有人指出 DES 现在就该废止），那么协议可以简单地使用其他密码系统，例如 IDEA 即可。

通过抽象而达到的一般性不仅仅是研究者的一个工具，也是实用密码协议开发的原则，比如 SSL 协议，这是一个嵌入在当前网页浏览器中并参与大多数安全 Web 交互的协议，这个协议的一般性设计，允许其可以随意使用密码模块的各种不同的实现。

对于大多数新的构建模块概念，例如公钥加密及数字签名，我们将探讨它

们在实践中采用的实现方式。但是，就像在前言中所说，对于其他一些构建模块——对称密钥加密、单向函数和消息摘要函数——我们将不再讨论它们在现实中典型的实现细节。这些构建模块的实现细节并没有多大启发性，构建这些模块的设计原则也超过了本书的讨论范畴。对于这些密码学构建模块，我们会讨论其他几种不同的实现方法，这些其他的实现方法如果运用正确的话，也能够和现实中使用的方案一样安全。我们将会仔细地提示读者，什么时候描述的方案是来自实践。

1.6　对密码系统的攻击

让我们通过关注最熟悉的密码学工具——加密，以使讨论变得稍微具体。假设 Alice 需要给 Bob 发送一条隐私信息，因此她将这个信息进行了加密，而密钥只有她和 Bob 知晓。窃听者 Eve 的目标在于侵犯通信过程的隐私。在最坏的情况下，Eve 可以准确地读取通信中的信息，这通常需要 Eve 知道加密过程所使用的密钥（回顾一下前言中卡萨诺瓦的故事）。但是，如果 Eve 通过监测密文获取了某些信息，虽然并没有获得准确和完整的通信信息，但这依然破坏了 Alice 和 Bob 通信过程的隐私性。即使一个密码系统像这样被部分破解，依然会产生严重的后果（例如我们后续将要学到的 VENONA 项目）。

“获得某些信息”是一种很模糊的说法，在本书的这一阶段，我们还不能从技术层面上对这一说法给出形式化的表述。无论如何，让我们来继续考虑在破解密码系统的过程中 Eve 能够利用哪些信息。通常区分四种类型的攻击，按照攻击强度开序排列，它们是：

- 唯密文攻击
- 已知明文攻击
- 选择明文攻击
- 选择密文攻击

假设通过截取 Alice 发送给 Bob 的一条或多条消息，Eve 能够对这些密文进行分析，从而获得关于这些密文所对应的明文的某些信息。这种情形下，我们说 Eve 使用了唯密文攻击。

假设 Eve 截取了一条密文，而她已经知道了这条密文所对应的明文。她可能会分析明文和密文之间的关系，以便得出关于密钥的某些信息。这些信息能够帮助她解密用相同密钥加密的其他信息。在这种情形下，我们说 Eve 采用了

已知明文攻击。在更一般化的这类攻击中，她或许能够获取和使用许多这样的明文-密文对。

下面我们考虑 Eve 采用的一种更为主动的攻击方式。假设她能够自己选择明文，并让 Alice 进行加密，而 Alice 无知地将 Eve 选择的明文进行加密。对 Eve 来说，能够获得由她选择的明文来生成的明文-密文对是非常有用的。上面我们描述的即是选择明文攻击。在历史上有过这种例子。美国在二战期间的一个驻日大使曾报告说："一个日本政府高官希望向我们政府发送一条密码消息，他们不希望这条消息被日本军方获知，因此他们让我将这条消息用我们最高级别的加密手段进行加密，我说当然我会这样做。"[⊖]

在 Eve 所能够进行攻击的一个变体中，她或许和 Bob 关系友好，能够让 Bob 将她选择的密文解密。具体而言，Eve 构造一个密文，而她对于这个密文所对应的明文一无所知，但是她能够说服 Bob 来向她提供该密文所对应的明
10 文。这便被称作选择密文攻击，事实上，这种攻击方式是很有效的，比如，当 Eve 构造的密文和她想要破解的密文在某种程度上具有相似性时。有一种针对 RSA 密码系统的攻击就是这种类型的。

1.7 思考题

1. 在下列每个通信场景中，请简述赞成或者反对该信道安全的理由。提示：无须给出技术层面的描述，对于这些问题，我们并没有特定的答案。
   (a) 通过电话下信用卡订单
   (b) 从自动取款机取现金
   (c) 通过邮件支付业务账单
   (d) 给你的教授发送一封电子邮件
2. 你注册了一门密码学课程，而你的死对头 Eve 恰好加入了这门课的教学团队。基于上述事实，请用自己的语言描述当你运行这门课程提供的软件时，
11 会面临什么样的风险。同样，你无须给出技术层面的答案。

⊖ Kahn，495 页。

第2章

A Cryptography Primer: Secrets and Promises

模 算 术

2.1 凯撒密码

我们知道（由当代作家苏维托尼乌斯所说[⊖]），尤里乌斯·凯撒利用了一种密码系统，该系统中的明文 A 被密文中的 D 所替换，B 被 E 所替换，以此类推。字母表最后三个字母分别被字母表前三个字母所替换。以下是该加密方案的一个例子：

明文	V	E	N	I	V	I	D	I	V	I	C	I
数值	21	4	13	8	21	8	3	8	21	8	2	8
+3	24	7	16	11	24	11	6	11	24	11	5	11
密文	Y	H	Q	L	Y	L	G	L	Y	L	F	L

为了从数学上描述凯撒密码，我们将字母表中的每个字母表示为 0 至 25 之间的某个整数：0 表示 A，1 表示 B，…，25 表示 Z。这样，加密 0（A）至 22（W）之间的某个明文元素只需要加 3 就可以了。那么，加密 23（X）、24（Y）和 25（Z）呢？对这三个元素，我们也对其加 3，只不过是以一种不同的方式——当超过 25 以后，我们接着从 0 重新开始。因此，对于 24，我们对其加 1，获得 25，然后再加 1，获得 0 而不是 26，然后再加 1，最终获得 1。同样，对于 25，我们加 1，获得 0 而不是 26，然后加两次 1，获得 2。

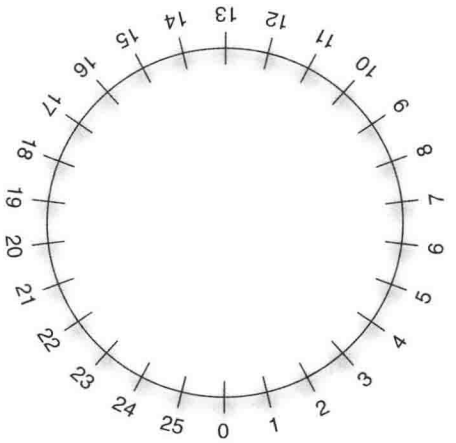
解密以相同的方式进行处理，但这里需要减去 3，而不是加 3——需要注意，0 减去 3 得到的是 23，1 减去 3 得到的是 24，2 减去 3 得到的是 25。 12

2.2 整数“圈”

正如可以将普通的加减法可视化为一个整数轴一样，我们也可以将上述这

⊖ 来自 Kahn，第 2 章。

种加减法可视化为一个整数圈：

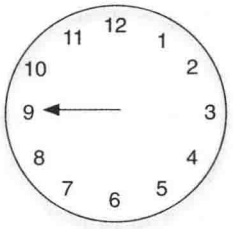


整数圈清楚地表明，如果沿从左至右的方向经过 25，则返回到 0；如果沿从右至左的方向经过 0，则到达 25。

利用该整数圈的加法称为*模 26 加法*。在模 26 的情境中，恰好有 26 个数字。我们在整数轴上将其标识为 0 到 25。数 26 称为模。当然，我们经常使用一些其他的模，在下面的章节中会有示例。

2.3 日常生活中的模算术

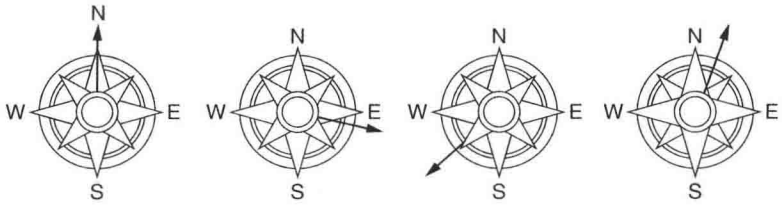
你之前肯定见到过模加法运算，虽然可能并不是以这个名字见到的。例如，考虑一个钟表的表盘：



假定现在是 9 点钟，你想知道 6 小时后是什么时间。这需要使用到模 12 算术：9 加 6 等于 3（模 12）。

在处理罗盘方位和角度时，用到的模数是 360。假定你现在正往正北方向走；方位是 0 度。然后向右转动 110 度；现在的方位是 110 度（大致是东南-偏

东方向)。然后再向右转动120度，方位变成230度。到目前为止，使用的都是角度的普通加法。然而，如果继续向右转动150度，则得到的方位是~~380~~ 20度。



考虑一周的每一天。我们将其表示为如下数字：

天	数字
周日	0
周一	1
周二	2
周三	3
周四	4
周五	5
周六	6

在该背景下，应该使用7作为模数。假定今天是周四，用数字4来表示。5天以后是周几呢？我们在4上5加得到~~9~~ 2，因此是周二。13天后是周几？我们在4上加13得到~~17~~ 3，所以是周三。700天以后是周几？注意到700天正好是100个周，因此和今天一样是周四。最后这个例子表明，加上7的倍数并不会带来结果的不同。加上7和加上14其效果是一样的，还有加上21，或者加上0，或者减去7。

2.4 同余

数学家利用模同余的概念来形式化模算术。我们称两个整数关于一个给定的模数同余，当它们相差模数的倍数。例如，如果模数是7，则3、10以及17是同余的。表述两个表达式同余的式子被称为同余式。 14

2.4.1 模7同余

我们用7作为模数。如果两个整数相差7的倍数，则这两个数模7同余。

例如，4 同余于 11（也就是 4+7）以及 18（也就是 4+2×7），甚至同余于−3（也就是 4+(−1)×7）。书写同余式的数学符号和书写等式的数学符号是相似的：等式符号（“=”）是由两条短线组成，而同余式符号由三条短线组成（“≡”）。例如，如下同余式

$$1 \equiv 8 \pmod 7$$

表示 1 和 8 关于模数 7 同余。注意在圆括号中我们用“modulo”的缩写“mod”来指明模数。

以下是更多关于模 7 的例子。读者可以通过检查这些数的差是否是 7 的倍数来验证它们是否真的同余：

$$2 \equiv 30 \pmod 7$$

$$-9 \equiv -2 \pmod 7$$

$$-20 \equiv 8 \pmod 7$$

$$7 \equiv 0 \pmod 7$$

$$14 \equiv 0 \pmod 7$$

当然，如果两个数相等，其差为 0，也是 7 的倍数（0 乘以 7 等于 0）。这说明任何真等式对应一个真同余式。例如，由于 4+5=9，所以同余式

$$4 + 5 \equiv 9 \pmod 7$$

成立。

最后，因为有时候反复地写“模 7”也比较麻烦，因此很多时候，当对于同一个模数执行很多次运算时，可以预先说明模数是什么，比如是 7，然后就不必再写明了。在每人都知道模数是 7 的情况下，可以直接说“1 和 8 同余”或直接写“1≡8”。

2.5 另一个例子：模 10 同余

本书作者最喜欢的模数之一是 10，因为当使用该模数时，很容易判断同余
15 式。比如说考虑整数 314 159。该数可以写成

$$314\ 159 = 31\ 415 \times 10 + 9$$

该等式表明，314 159 与 9 相差 10 的倍数。因此下面的同余式成立：

$$314\ 159 \equiv 9 \pmod{10}$$

类似地，任何正整数都与它的个位数（模 10）同余，因为剩下的整数是 10 的倍数。

由此推断，如果两个正整数个位数相同，则二者（模 10）同余。

2.6 同余代换

我们应该知道如何将子表达式替换成与之相等的子表达式来简化等式。例如，由于 12+4 等于 16，因此，我们可以将等式

$$(12+4)\times 2 = x+1$$

写成

$$16\times 2 = x+1$$

在处理同余式时，也可以将涉及加法、减法和乘法的子表达式替换为与其同余的子表达式。这里有一个例子，假定下面的模 7 同余式成立：

$$(12+4)\times 2 \equiv x+1 \pmod 7$$

由于 12+4（模 7）同余于 2，我们有

$$2\times 2 \equiv x+1 \pmod 7$$

同余代换的思想非常重要，因此我们将其表述成如下原理：

同余代换原理 在任意一个成立的模同余式中，将任意子表达式替换为一个与其同余的子表达式，所得的模同余式仍成立。

16

2.6.1 使用代换简化多个数相加

代换可以使得模运算更加简单。我们从如下同余式开始：

$$5+6+3+4+5+6 \equiv x \pmod 7$$

因为 5+6 等于 11，而 11 与 4 同余，因此可以将上述式子中的子式 5+6 替换为 4，这样得到

$$4+3+4+5+6 \equiv x \pmod 7$$

继续这一方法，4+3 等于 7，而 7 与 0 同余，因此可以将子式 4+3 替换为 0：

$$0+4+5+6 \equiv x \pmod 7$$

因为 0+4+5 等于 9，而 9 与 2 同余，将 0+4+5 替换为 2，得到：

$$2+6 \equiv x \pmod 7$$

最后，2+6 等于 8，8 与 1 同余，所以有

$$1 \equiv x \pmod 7$$

2.6.2 使用代换简化多个数相乘

在有乘法运算的情况下，利用代换可以防止数变得过大，因此该技术更加有用。我们从如下同余式开始：

$$5 \cdot 6 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \equiv x \pmod 7$$

因为 5 · 6 等于 30，而 30＝4 · 7＋2，因此 30 同余于 2，因此我们可以将子式 5 · 6 替换为 2，得到：

$$2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \equiv x \pmod 7$$

继续，2 · 3 等于 6，所以有

$$6 \cdot 4 \cdot 5 \cdot 6 \equiv x \pmod 7$$

因为 6 · 4 等于 24，同余于 3，我们将 6 · 4 替换为 3，得到

$$3 \cdot 5 \cdot 6 \equiv x \pmod 7$$

因为 3 · 5 等于 15，同余于 1，我们将 3 · 5 替换为 1，得到

17

$$1 \cdot 6 \equiv x \pmod 7$$

也就是

$$6 \equiv x \pmod 7$$

在上述推导中，我们不需要乘比 6 大的数，因为在每次乘法之后，我们将乘积替换为比模数小的同余数了。这个技巧在我们处理同余式中涉及数以千计的整数乘法时是很必要的。在此后的章节中可以发现，这种同余式会在密码学中出现。

2.6.3 舍九法

模九算术是检查一般的算术是否正确的一种有效的技巧。假定你进行了一次涉及乘法和加法的运算，但你并不能确定计算结果是否正确。例如，下面的等式成立吗？

$$5837 \cdot 48 + 42\,090 \overset{?}{=} 327\,066$$

正如 2.4.1 节中所说，如果等式成立，则对任意模数，相应的同余式也成立。特别地，使用 9 作为模数，如果上述等式成立，则下面的同余式也应该成立：

$$5837 \cdot 48 + 42\,090 \overset{?}{\equiv} 327\,066 \pmod 9$$

为了简化该同余式，我们将 5837 写成十进制形式：

$$5837 = 5 \cdot 10^3 + 8 \cdot 10^2 + 3 \cdot 10 + 7$$

- 首先考察 $3 \cdot 10$ 模 9，因为 10 同余于 1，所以 $3 \cdot 10$ 同余于 $3 \cdot 1$，也就是同余于 3。
- 其次考察 $8 \cdot 10^2$，10^2 等于 $10 \cdot 10$。因为 10 同余于 1，所以 $10 \cdot 10$ 同余于 $1 \cdot 1$，也就是 1，因此，$8 \cdot 10^2$ 同余于 $8 \cdot 1$，也就是 8。
- 最后考察 $5 \cdot 10^3$，同样，10^3 同余于 1，因此 $5 \cdot 10^3$ 同余于 5。

将上述结果合在一起，$5 \cdot 10^3 + 8 \cdot 10^2 + 3 \cdot 10 + 7$ 同余于 $5+8+3+7$，也就是说，5837 同余于它每位数字之和。

上述过程意味着对于任意正数：由于 10、10^2、10^3 以及 10 的更高幂次都能（模 9）同余于 1，因此任意正整数都（模 9）同余于它每位数字之和。 18

可以反复使用这种思想。我们看到 5837 同余于 $5+8+3+7$，也就是 23。而 23 也同余于它本身每位数字之和 $2+3$，也就是 5。因此，5837 与 5 同余。

那 $5837 \cdot 48$ 呢？48 同余于它每位数字之和 $4+8$，也就是 12。12 同余于它每位数字之和，也就是 3。因此，通过代换，$5837 \cdot 48$ 同余于 $5 \cdot 3$，也就是 15，而 15 同余于 $1+5$，也就是 6。

$5837 \cdot 48+42\,090$ 呢？我们已经知道 $5837 \cdot 48$ 同余于 6。根据“数位和”原则，我们可以知道 42 090 同余于 $4+2+0+9+0$，也就是 $6+9$。因为我们正在进行模 9 运算，这样 9 同余于 0，所以 $6+9$ 同余于 6。（任何时刻只要我们看到 9，就可以在数位求和中省去；任何时刻只要我们看到某些数位上的数加起来是 9，比如 2 和 7，也可以直接省去。这就是该技术被称为“舍九法”的原因。）

我们知道 $5837 \cdot 48$ 同余于 6，42 090 同余于 6。因此，通过代换，$5837 \cdot 48+42\,090$ 同余于 $6+6$，也就是 12，由于 12 每位数字的和为 3，因而同余于 3。

我们的目的是检查以下同余式是否正确：

$$5837 \cdot 48 + 42\,090 \stackrel{?}{\equiv} 327\,066 \pmod 9$$

式子左边同余于 3。式子右边呢？一种简化右边的方法是将每位数字加起来，得到 24，然后将 24 的每位数字加起来，得到 6。一种更快的方式是注意到 2 和 7 加起来得 9，3 和 6 加起来得 9，剩下的数字是 0 和 6，加起来是 6。这两种方式都表明，式子右边同余于 6。此前，我们已经知道式子左边同余于 3。因为 3 并不和 6 同余，所以该同余式不成立。这也就说明此前的等式 $5837 \cdot 48+$

42 090＝327 066 不成立。

我们阐述了一种检查涉及加法和乘法的普通运算是否成立的方法。这种检查方法与直接算出所有加法和乘法相比更容易，也更不易于出错。该方法也不能保证万无一失。如果等式并不成立，但是左边和右边相差 9 的倍数，则相应的同余式也成立。然而，大多数情况下，该方法还是有效的。对我们来说，更
19 重要的是，这种方法说明，在处理同余式时使用代换是很有用的。

2.7 代表元与余数

考虑一个代议制民主制度，其中每个人有且仅有一个代表。我们在模算术中引入这个思想。对于一个特定的模数 m，我们定义整数 0，1，2，…，$m-1$ 为代表元。这一定义来源于下面的定理：

代表元定理：*每个整数恰好和整数 0，1，2，…，$m-1$ 中的一个模 m 同余。*

例如，在模 7 的情况下，每个整数和整数 0，1，2，3，4，5，6 中的一个同余。

2.7.1 商和余数

上述定理基于数论中的一个基本结论：

带余除法定理：*对每个整数 b 和每个正整数 m，存在唯一整数 q 及唯一整数 r，其中 r 属于 0，1，2，…，$m-1$，满足*

$$b = qm + r \tag{2.1}$$

例：

- 令 $b=25$，$m=7$，则等式（2.1）在 $q=3$，$r=4$ 时成立（即 $25=3\cdot 7+4$）。
- 令 $b=62$，$m=7$，则等式（2.1）在 $q=8$，$r=6$ 时成立（即 $62=8\cdot 7+6$）。
- 令 $b=99$，$m=12$，则等式（2.1）在 $q=8$，$r=3$ 时成立（即 $99=8\cdot 12+3$）。

如上面例子所示，当 b 被 m 除时，r 称为余数，q 称为商。计算余数和加法、乘法一样，也是一种算术运算。鉴于加法、乘法都有各自的运算符＋和×，我们也为余数操作定义一个运算符 rem。我们用 b rem m 来表示带余除法定

理中的 r。（rem 是英文 remainder 的简写。）

例：

- 25 rem 7 是 4。
- 62 rem 7 是 6。
- 99 rem 12 是 3。

那么 rem 与模算术有什么关系呢？事实上，b rem m 的值恰恰就是 b 模 m 的代表元！ 20

- $r=b$ rem m 是整数 0，1，2，…，$m-1$ 中的一个，也就是说它是代表元。
- 等式 $b=qm+r$ 表明 b 和 r 相差 m 的倍数，也就是说 b 同余于 r（mod m）。

2.7.2 利用 rem 检查两个数是否同余

rem 的一个用途就是判断两数 a 和 b 是否模 m 同余。

判定同余关系

我们可以使用下面的论证。

> 假定 a 和 b 具有相同的模 m 代表元，因而它们都与该代表元（模 m）同余，从而二者彼此同余。

数学上，我们写作

> 如果 a rem $m=b$ rem m，则 $a\equiv b$ (mod m)。

30（mod 7）同余于 51 吗？为了回答这个问题，我们计算 30 和 51 的模 7 代表元。30 除以 7 的余数是 2，因此 2 是 30 的代表元。51 除以 7 的余数是 2，所以 2 也是 51 的代表元。因此，30 同余于 2，51 也同余于 2，所以 30 和 51 彼此同余。

判定非同余关系

可以利用 rem 来判定两个整数是否同余。那么，可以利用 rem 来判定两个整数是否不同余吗？

我们使用下面的论证。

假定 a 和 b（模 m）同余。因为 a 同余于其代表元，b 同余于 a，所以 b 同余于 a 的代表元。因为 b 仅和整数 0，1，2，…，$m-1$ 中的一个同余，即和它的代表元同余，所以 b 的代表元就是 a 的代表元。

21

数学上，我们写作

如果 $a \equiv b \pmod{m}$，则 $a \text{ rem } m = b \text{ rem } m$。

这就表明，如果 a 的代表元不同于 b 的代表元，则 a 和 b 不同余。

40（mod 7）同余于 73 吗？我们计算 40 和 73 的代表元。40 除以 7 的余数是 5，所以 5 是 40 的代表元。73 除以 7 的余数是 3，所以 3 是 73 的代表元。由于 40 和 73 具有不同的代表元，因此它们不同余。

2.7.3 使用 rem 简化模同余式

在 2.6.1 节和 2.6.2 节中，我们使用同余代换原理来判定复杂的同余式是否成立。每一步我们将子表达式替换为与其同余的子式。如果采用这样的法则，即总是将子表达式替换为其代表元，我们就可以保证计算中间结果不会变得过大——特别地，可以保证中间结果总是小于模数。

假定我们想要解同余式

$$12 \cdot 11 \cdot 10 \cdot 9 \cdot 8 \cdot 7 \equiv x \pmod{13}$$

在计算左边时，可以不将所有乘法全部算完，而是每次仅进行一次乘法，并用其模 13 的代表元替换乘积：

12 · 11 等于 132，而 132 除以 13 的余数是 2，所以我们用 2 来替换 12 · 11，这样同余式变成

$$2 \cdot 10 \cdot 9 \cdot 8 \cdot 7 \equiv x \pmod{13}$$

2 · 10 等于 20，20 除以 13 的余数是 7，所以我们将 2 · 10 替换成 7，获得同余式

$$7 \cdot 9 \cdot 8 \cdot 7 \equiv x \pmod{13}$$

进一步地，我们将 7 · 9 替换为 7 · 9 rem 13，也就是 11，获得

$$11 \cdot 8 \cdot 7 \equiv x \pmod{13}$$

然后将 11 · 8 替换为 11 · 8 rem 13，也就是 10，得到

22

$$10 \cdot 7 \equiv x \pmod{13}$$

将 $10 \cdot 7$ 替换为 $10 \cdot 7 \text{ rem } 13$，也就是 5。最后得到

$$5 \equiv x \pmod{13}$$

因此，$x=5$ 是一个解。

2.7.4 利用 rem 简化涉及 rem 计算的等式

假定要求计算

$$12 \cdot 11 \cdot 10 \cdot 9 \cdot 8 \cdot 7 \cdot 6 \text{ rem } 13$$

我们知道，计算余数得到的结果就是模 13 的代表元。根据以下事实：

$$\text{如果 } a \equiv b \pmod{m}, \text{则 } a \text{ rem } m = b \text{ rem } m$$

我们可以看到，如果将 $12 \cdot 11 \cdot 10 \cdot 9 \cdot 8 \cdot 7$ 替换为任何与其同余的子式，结果不变。2.7.3 节中的运算结果表明

$$12 \cdot 11 \cdot 10 \cdot 9 \cdot 8 \cdot 7 \equiv 5 \pmod{13}$$

因此，我们只需要计算

$$5 \cdot 6 \text{ rem } 13$$

很容易得到表达式的计算结果为 4。

2.7.5 负整数的代表元

我们已经知道，$b \text{ rem } m$ 是 b 除以 m 的余数。但是，我们所说的余数在带余除法定理中定义为非负整数。当 b 是负数时，这会跟读者关于余数概念的直觉相冲突。例如，有读者可能会认为 $-15 \text{ rem } 7$ 的结果是 -1（-15 除以 7，商为 -2，余数为 -1），但是我们需要结果是一个模 7 的代表元。同余于 -1 的代表元是 6，所以我们定义 $-15 \text{ rem } 7$ 的结果是 6。同样，$-3 \text{ rem } 7$ 等于 4，$-25 \text{ rem } 7$ 等于 3。

2.8 思考题

1. 给出下列问题的答案并说明原因。今天是周三。
   (a) 7 天以后是周几?
   (b) 67 天以后是周几?
   (c) 如果一个获得通过的政府提案决定，在周六后面加上一天，那么从今天开始算，67 天以后是周几?

2. 给定数 4.512 835 128 351 283 512 83…，小数点后第 295 位数是多少？并给

出你答案的原因。

3. 一个简单的两人游戏是这样的：你和你的对手面前有一堆硬币。你们轮流拿硬币，可以拿一枚、两枚或者三枚。你的目的是让对手拿到最后一枚硬币。和朋友试试这个游戏，然后回答下列问题。

(a) 请补充完整下面的表格。

硬币数目	最佳策略
1	失败
2	拿1枚
3	拿2枚
4	拿3枚
5	?
6	?
7	?
8	?
9	?
10	?

(b) 令 n 为桌面上的硬币数目。现在轮到你拿硬币。请问你需要拿几枚？利用 n 给出你的答案。

(c) 考虑这个游戏的变体——每个玩家可以拿一枚、两枚、三枚或者四枚硬币。请补充完整下面的表格：

硬币数目	最佳策略
1	失败
2	拿1枚
3	拿2枚
4	拿3枚
5	拿4枚
6	?
7	?
8	?
9	?
10	?

24 (d) 对于问题的变体，请回答问题 (b)。

4. 对于以下同余式，求出解 x。你给出的解应该是代表元（当模数是 m 时，应是 0，1，2，…，$m-1$ 中的一个）。

(a) $x \equiv 4+3 \pmod 8$

(b) $x \equiv 7+34 \pmod 4$

(c) $13+22\equiv x \pmod{13}$

(d) $6-7\equiv x \pmod{88}$

(e) $x\equiv 9-19 \pmod{7}$

5. 给出下列同余式的解（代表元）。

(a) $12+4+7+12+17\equiv x \pmod{12}$

(b) $7+x\equiv 2 \pmod{3}$

(c) $x\equiv 16+2 \pmod{8}$

(d) $x\equiv 57\cdot 73 \pmod{17}$

(e) $x\equiv 3\cdot(87-5) \pmod{7}$

(f) $x\equiv(-5)-(-11) \pmod{10}$

6. 计算：

(a) $16+2$ rem 8

(b) $57\cdot 73$ rem 17

(c) $3\cdot(87-5)$ rem 7

(d) $(-5)-(-11)$ rem 10

第3章
A Cryptography Primer: Secrets and Promises

加法密码：一个不安全的分组密码

凯撒密码系统[⊖]可以被轻易地攻破。最直接的攻击利用了凯撒密码的密钥可选取范围过小这个事实。假设攻击者 Eve 截取到如下密文：

UMTSJENXEYFUUJI

该密文使用凯撒密码系统加密，所使用的字母表包含了空格与 26 个英文字母 A，…，Z。Eve 首先可以枚举所有 27 种可能的密钥，并找出密文在每个密钥下对应的明文，从而得到[⊜]：

UMTSJENXEYFUUJI	
TLSRIDMWDXETTIH	GZFEWR JRKSGGWV
SKRQHCLVCWDSSHG	FYEDVQZIQJRFFVU
RJQPGBKUBVCRRGF	EXDCUPYHPIQEEUT
QIPOFAJTAUBQQFE	DWCBTOXGOHPDDTS
PHONE#IS#TAPPED	CVBASNWFNGOCCSR
OGNMDZHRZS#OODC	BUA#RMVEMFNBBRQ
NFMLCYGQYRZNNCB	AT#ZQLUDLEMAAQP
MELKBXFPXQYMMBA	#SZYPKTCKDL##PO
LDKJAWEOWPXLLA#	ZRYXOJSBJCKZZON
KCJI#VDNVOWKK#Z	YQXWNIRAIBJYYNM
JBIHZUCMUNVJJZY	XPWVMHQ#HAIXXML
IAHGYTBLTMUIIYX	WOVULGPZG#HWWLK
H#GFXSAKSLTHHXW	VNUTKFOYFZGVVKJ

如果 Eve 有理由相信明文是英文句子，她应该不难从上述可能的明文列表中猜出消息内容。

这个例子说明如果一个密码系统的密钥取值范围过小，则该系统是不安全的。然而，这个结论的逆命题并不成立：一个密钥取值范围足够大的密码系统未必是安全的。在本章中，我们给出一个简单的密码系统，它是一个一般化的

⊖ 这里说的应该是比前面提到的凯撒密码系统更一般的情况，加密规则是将每个字母用其后面的第 k 个字母代替；前面提到的凯撒体制是 $k=3$ 的情形。——译者注

⊜ 为便于读者观察，我们调换了解密结果的顺序，解密结果按 $k=0$，1，2，…，26 的顺序列出，并用 # 代替空格。——译者注

凯撒密码，我们称之为加法密码。我们以该系统为例说明分组密码的概念以及一些简单的密码分析攻击。

3.1 加法密码

我们以凯撒密码作为起点，并尝试修复它最明显的缺陷：密钥取值范围过小。加密的方法使用模加运算，其中的模数为字母表的大小，并且每个代表元都对应一个密钥。我们在此大幅度地增大模的取值，这样将增大密钥取值范围。例如，可以令模数取值为 1000000000000（即 10^{12}）。想象在一个含有 10^{12} 个字母的外星字母表上使用凯撒密码，密钥将是与该模数相对应的一个代表元，即从 0 到 999999999999 中的任意一个数字。要加密这个外星字母表的一个字母，我们可以将字母对应的数字与密钥相加，并把和约减为代表元（即小于 10^{12} 的非负整数），这就是密文的一个字符。

我们想要在明文取自我们自己的小字母表时使用这个加密系统。这里有一种方法：使用 0 到 999999999999 中的数去表示由 6 个符号组成的序列，数的每两位表示一个符号。两位数字可以表示 100 个符号，这对于表示如空格、小写字母、大写字母、数字以及各种印刷符号来说绰绰有余了。例如，字符串“we try”可以表示成数字 230500201825。假如 Alice 希望可以使用密钥 620487370109 加密这个消息。通过计算 230500201825 加 620487370109 模 10^{12}，我们可以得到密文 850987571934。

3.2 分组密码

显然，一个只能加密 6 个符号组成的明文的加密系统的用处非常有限。那么 Alice 应如何使用加法密码系统加密包含更多符号（例如，50 个符号）的明文呢？这类方案一般的名称叫作*分组密码*（block cypher），基本的加密方法称为*分组加密*（block encryption），分组加密可以处理的明文大小称为*分组长度*，分组长度可以通过明文的位数来度量。例如，在 3.1 节所描述的加密系统中，分组长度为 12（十进制）位，分组加密方法是将明文与密钥进行模 10^{12} 的加法运算。

为使一个分组密码能处理任意长度的明文，最简单的方案称为电子密码本（Electronic Code Book，ECB）模式，如图 3.1 所示。Alice 先将明文消息分组。 27
第一个分组由前 12 位组成，第二个分组包括接下来的 12 位，以此类推。在这

个例子中一共有五个这样的分组，最后一个分组使用额外的 0 进行填充使其保持正确的分组长度。下一步 Alice 分别加密每一个分组，所有的分组使用相同的密钥加密。最终密文为每个分组加密所得到的密文组成的序列。

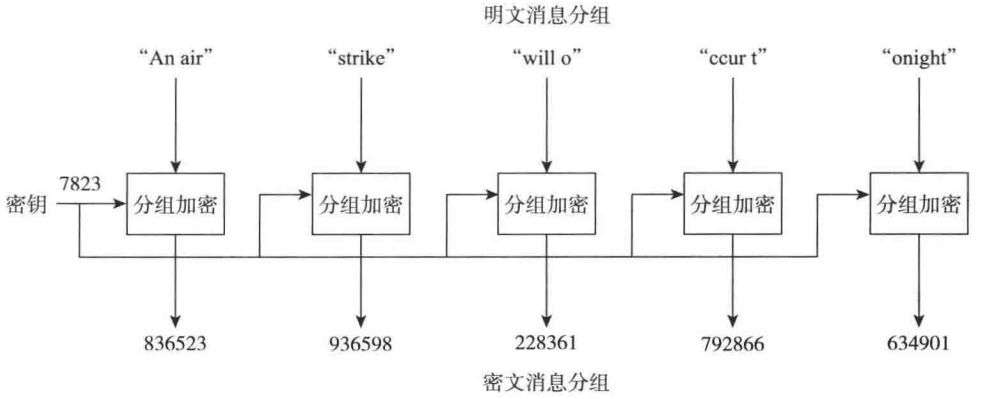


图 3.1 ECB模式示意

下面让我们看一个小例子。假如 Alice 使用一个仅支持两位明文的加法密码系统，即明文取值范围为 0 到 99 之间的数字，系统的密钥也是 0 到 99 之间的数字。密文通过将明文与密钥模 100 相加产生，所以密文也是 0 到 99 之间的数字。假设 Alice 想要加密一个 16 位的明文，例如 3057205711654928。假如她的密钥为 11。那么她首先将明文分成八个分组：

30 57 20 57 11 65 49 28

接着，她使用她的密钥 11 分别加密每一个分组。对 30 的加密为 41，对 57 的加密为 68，以此类推。最终她获得八组密文：

41 68 31 68 22 77 60 39

她把这些密文分组合并起来，得到密文 4168316822776039，并传给 Bob。要解密该密文，类似地，Bob 将密文分成八个分组，并依次解密每一个分组。

值得注意的是，上述的方案本质上是一个凯撒密码系统（使用模数 100 代替凯撒密码中的 26）。如果明文含有某些可以预测的特性，则密文会有一些相关的性质。例如，如果分组 05 代表字母 e，那么它会在明文中频繁出现，因此
28 分组 16 将会在密文中频繁出现。如果 Eve 知道所使用的密码系统有这种特点，但是不知道密钥，那么她可以分析一段长的密文并有望借此获得关于密钥的信息，进而获得关于明文的信息。为了抵抗这种类型的攻击，设计分组密码的密码学者们应该确保分组长度足够大，以至于这种模式很难发生，比如分组长度

为 20 位甚至更多。

3.3 对加法密码的攻击

即使分组长度非常大，使用电子密码本模式的加法密码系统仍不安全。我们讨论一些对它的攻击。

3.3.1 已知明文攻击

假设 Alice 和 Bob 已预先协商了一个密钥，并且 Alice 向 Bob 发送了一个利用加法密码系统加密的长消息，其分组长度为 20 位数字，进一步假设 Eve 截获了密文而且她碰巧知道或者猜测 Alice 在给 Bob 的每段消息的开头都会写“Dear Bob”。因此 Eve 知道了第一个分组的明文和密文。她利用分组加密公式

$$cyph = plain + key \text{ rem } 10^{20}$$

来计算密钥：

$$key = cyph - plain \text{ rem } 10^{20}$$

Eve 现在可以利用计算出来的密钥解密密文中剩下的所有分组了。

我们可以看到，如果 Eve 知道明文中的一个分组以及对应的密文，则她可以将密钥求出。

接下来，假设 Eve 并不确切地知道 Alice 消息的开头，她只知道 Alice 消息开头的常见形式，或者说只知道 20 种（甚至 10 000 种）可能的方式，她都可以利用上述过程，为每一种可能的第一个明文分组算出对应的密钥。然后她依次尝试用每一个密钥去解密整个消息。对每一个错误密钥（证明是不正确的候选密钥）来说，Eve 很有可能得到无意义的明文。对正确的密钥来说，Eve 会得到有意义的明文。

当然，对于某些错误密钥来说，得到的明文同样可能并非胡言乱语。但是，当消息足够长的时候，这种事情是极其罕见的。退一步来说，即使 Eve 仅仅是 29
将 Alice 所发送的可能的明文范围缩小到两三个候选明文之中，她同样成功地威胁了 Alice 的隐私，并因此攻破了方案的安全性。值得注意的是，在后一种攻击中，Eve 利用了她对 Alice 可能发送的消息类型的先验知识进行攻击。具体来说，她利用了她对明文第一个分组的相对精确的知识（这种技巧可以针对任何特定的分组，例如最后一个分组），以及利用了整个消息非常有可能是英文语言这一事实。

3.3.2 唯密文攻击

在有些情形中，为了从密文中获得有用的信息，Eve甚至不需要对明文有任何的了解。此时，她可以利用对消息固有的一般知识进行攻击。

例如，假设在每月初，Alice都会向Bob发送一个消息，该消息包含Bob在本月为Alice的产品做广告的预算费用。该消息利用20位数字的加法密码系统进行加密，并且每月都使用相同的密钥。（即使Alice为比尔·盖茨工作，假设明文不会超过10^{20}也是不会有问题的。）Eve是Alice的竞争者，并且想要知道Alice每月广告预算的变化。假设Eve截获了一月和二月的密文。她得到了以下两个等式：

$$Jan.\ cyph. = Jan.\ plain + key \text{ rem } 10^{20}$$

$$Feb.\ cyph. = Feb.\ plain + key \text{ rem } 10^{20}$$

注意到通过将两式相减，她得到：

$$Jan.\ cyph. - Feb.\ cyph. \equiv Jan.\ plain - Feb.\ plain \pmod{10^{20}}$$

由于Eve知道一月和二月的密文，她可以通过进行模10^{20}的减法获得一月和二月明文模10^{20}的差。也就是说，她可以获得一月与二月广告预算的差值模10^{20}的代表元。

真正的差值或者是上述差值模10^{20}的代表元（如果真正的差值为正数的话），或者上述差值模10^{20}的代表元减去10^{20}（如果真正的差值为负数的话）。Eve应该不难看出哪个是真正的差值。（对于读者来说，尝试举一个小数目的例
30 子可能更清楚，例如预算大概在10或者15左右，模大小为50。）

3.4 对使用ECB模式的分组密码的攻击

目前为止，我们已经看到了如何利用加法密码系统的特性进行攻击。然而，任何使用ECB模式的分组密码系统都有安全性缺陷。例如，注意到如果在不同的消息中出现同样的明文分组，那么其对应的密文分组也相同。假设Alice给她的代理人Bob发送经过加密的买卖订单。Eve截获了这些密文，然后等着看Bob会随后在市场中采取何种行为。如果在后面的密文中重新出现了某些相同的分组，Eve就可以对某些订单做出一些猜测，并可以针对Bob的行动做出一些针对性的布置。

由于ECB模式的安全性缺陷，它很少被使用。我们会在随后的章节中讨论

一些更加安全的分组加密模式。

3.5 思考题

1. 假设你是窃听者 Eve。你截获了如下密文符号："L"。你知道它是使用凯撒密码系统进行加密得到的，但是你不知道密钥。试列举出所有的密钥，并确定所有可能的解密结果。
2. 现在考虑将明文/密文符号表上的加/解密函数作为分组密码的基础。即单独加/解密明文/密文的每一个符号。假如你还是 Eve，并且截获了如下使用基于凯撒密码系统的分组加密方案所加密的密文。假如明文为一段英文，应该怎么解密这个消息呢？

   KYVVCVGYREKZJREXIP

3. 现在假设你已经知道密钥为 17 了，明文是什么？
4. 假如你正使用一个只含有四个符号的字母表（例如，字母"A" "B" "C" "D"）。
   (a) 在这个字母表上可以使用多少种不同的凯撒密码系统？（也就是说，密钥空间有多大？）
   (b) 该字母表上有多少种不同的代换密钥？

第4章
A Cryptography Primer: Secrets and Promises

函　数

4.1　基础知识

一个*二元关系*是一个集合中成员与另一个集合中成员的配对方式。我们可以使用图示来表示一个关系：对于关系中形成的每个对子，存在一个从该对成员中的第一项指向第二项的箭头。因此本章中从图 4.1 开始的所有的图均表示二元关系。

如果在关系中存在一个箭头 $x \rightarrow y$，我们说“x 映射到 y”，且在这关系下“y 是 x 的像”。因此在图 4.1 描述的关系中，元素 1，4 和 5 都映射到 96，1 还映射到 94，元素 2 映射到 100，最后 3 映射到 99。另一种说法是 96 是 1 的像，同时也是 4 和 5 的像，其他类似。

一个*单输入函数*是一种特殊的二元关系，其中第一个集合中的每一项恰好有一个出发的箭头，也就是说第一个集合中的每一项恰好映射到第二个集合中的一个元素。因此图 4.2 表示一个单输入函数，但是图 4.1 不是。实际上，图 4.1 描述的关系不是一个函数，有两个原因：1）第一个集合中存在元素（即元素 1）映射到两个元素，2）存在没有映射到其他元素的元素（即元素 6）。

第一个集合被称为函数的*定义域*，第二个集合被称为*上域*（或*陪域*，codomain）。因此图 4.2 所描述的函数的定义域是 {0，1，2，3，4，5}，上域是 {0，1，2，4，6，12，20}。注意到对于这个函数，并不是上域中的每一个元素都是某个元素的像（也就是有箭头指向它）。换言之，上域中存在元素（即 1 和 4）没有进入的箭头。这是可以的，它并不违背函数的定义。（稍后我们将讨论这样的函数，其上域中的每一个元素至少有一个进入箭头，这样的函数被称为*映上的*，即*满射*。）上域中所有定义域中元素的像组成函数的*值域*。因此一个函数的值域是其上域的子集合。图 4.2 中函数的值域是 {0，2，6，12，20}。

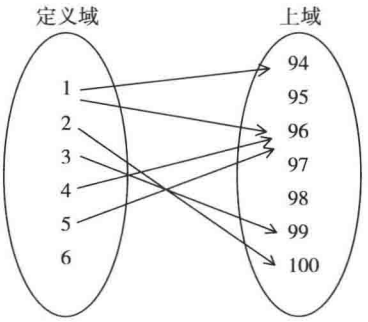


图 4.1　一个不是函数的二元关系

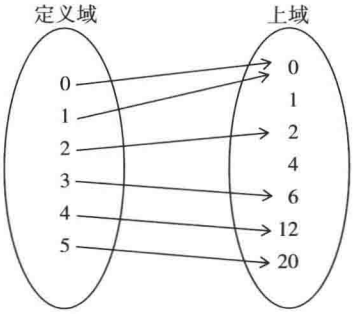


图 4.2　一个函数例子，其定义域是 {0,1,2,3,4,5}

考虑将定义域中的元素作为函数的输入。相对应的输出是输入所指向的元素。我们指出输出的集合即为值域。再一次强调，函数的最主要法则是**定义域中的每一个元素必须有单一的输出箭头**。

对于某些函数，存在一个很好的数学法则告诉你，对每一个输入，如何计算相应的输出。例如，对于图 4.2 描述的函数，该法则是

$$x \mapsto x \cdot (x-1)$$

利用这一点，假设你想计算与输入 3 相对应的输出。复制↦右边的公式，但是用 3 取代 x。产生的结果告诉你与 3 相对应的输出（即 3 • 2，结果为 6）。

然而，你必须谨记以下两点。

1. 法则并不能完全确定一个函数；定义域和上域也必须被指定。

2. 并不是所有的函数都有很好的法则去描述。

33

第 1 点并没有严格地被遵守，因为定义域和上域经常在上下文中是非常明显的。例如，如果计算模 7，则定义域和上域很可能就是 {0,1,2,3,4,5,6}。

4.2 可逆性

图 4.3 描述了另外一个函数，它与上一个例子有相同的定义域。

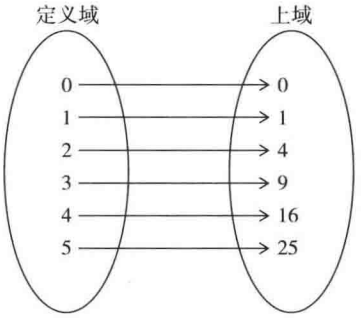


图 4.3 定义域与图 4.2 相同的函数

这个例子与上面的例子有一点重要的不同，在这个例子中，上域中的每个元素恰好是定义域中一个元素的像（定义域中有且仅有一个元素以这个元素为像）。很容易从图示中识别这类函数。（这是我们使用这类图示的主要原因。）

这一特性的意义在于，这样一个函数是*可逆的*。也就是说，如果把箭头反转，把以前的值域当作定义域，把以前的定义域作为上域，你会得到一个新函数。这一新函数被称为旧函数的*逆*。因此图 4.3 中函数的逆是图 4.4 所描述的函数。

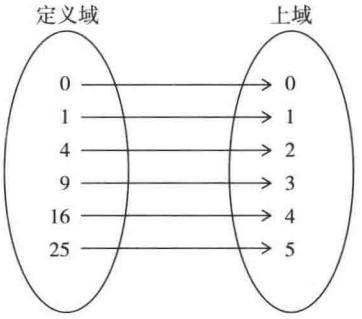


图 4.4 图 4.3 中函数的逆函数

尝试着对图 4.2 中的函数做同样的处理，你会得到图 4.5 中描述的二元关
34 系。注意到新定义域中的某些元素（例如 0）有多条箭头出去，因此该图不能表示一个函数。

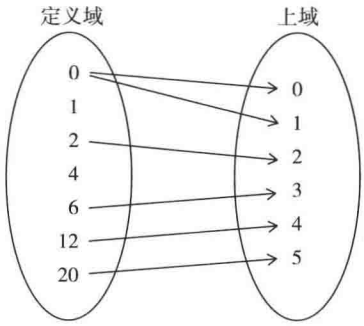


图 4.5　对图 4.2 中的函数尝试给出其逆的表示，该关系不是一个函数

回到可逆函数及它们的逆。猜想一下，图 4.4 所描述的函数的逆是什么？是的，它的逆函数就是图 4.3 中的函数。一个函数的逆的逆是原来的函数。（这里没什么深奥的；你把箭头反转一次，然后将它们再反转一次。它们最终回到开始的状态。）

当我们知道一个可逆函数的法则时，有时候可以使用该法则了解函数逆的意义。例如，图 4.3 所描述的函数的法则是 $x \mapsto x \cdot x$。也就是说，该法则说的是将输入平方以得到输出。平方的相反（形式化地，是逆）操作是什么呢？平方根！因此逆函数的法则（即图 4.4 中的函数）是 $y \mapsto \sqrt{y}$（在法则中我们使用一个不同的变量 y，这实际上并没有什么影响）。

图 4.6 描述了一个函数。描述该函数的一个法则是 $x \mapsto x+2 \text{ rem } 6$。该函数将 0 映射到 2，1 映射到 3，以此类推。因此函数的逆应该将 2 映射到 0，将 3 映射到 1，等等。逆函数在图 4.7 中显示。描述该逆函数的一个法则是 $y \mapsto y+4 \text{ rem } 6$。

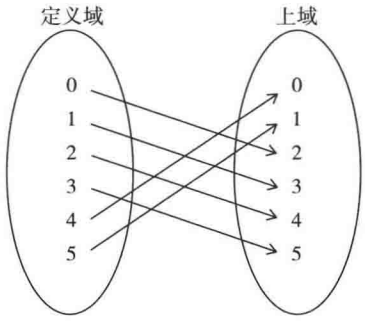


图 4.6　该函数的定义域为 {0，1，2，3，4，5}，法则是 $x \mapsto x+2 \text{ rem } 6$

35

这里有另一个例子。在图 4.8 中描述了一个函数。我们可以使用法则 $x \mapsto x+3 \text{ rem } 6$ 来描述该函数。

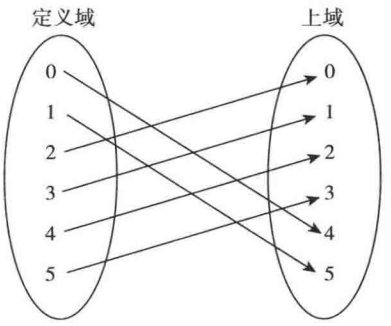


图 4.7 图 4.6 中函数的逆函数

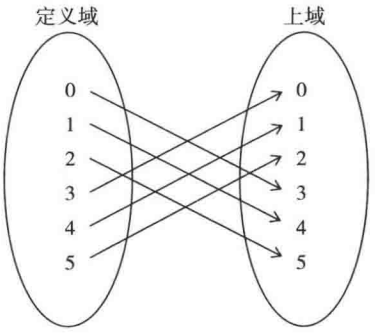


图 4.8 该函数的定义域为 {0，1，2，3，4，5}，法则是 $x \mapsto x+3 \text{ rem } 6$

该函数的逆是什么呢？该函数将 0 映射到 3，1 映射到 4，2 映射到 5，…，因此逆函数应该将 3 映射到 0，4 映射到 1，5 映射到 2，…。等一下，该函数是它自己的逆！

4.2.1 一对一和映上

我们已经说过，如果一个函数的每个上域中的元素都恰好是定义域中某一个元素的像，则该函数是可逆的。让我们将这一条件分解成两个条件，即一对一和映上：

- 一个函数是一对一的，如果该函数上域中的每个元素是定义域中**至多**一个元素的像。
- 一个函数是映上的，如果该函数上域中的每一个元素是定义域中**至少**一个元素的像。

36

这些术语被用来解释为什么一个函数不是可逆的。一个不是一对一的函数是不可逆的，同样，一个不是映上的函数也是不可逆的。（一个函数如果既不

是一对一的也不是映上的，则必然不是可逆的。）

另一方面，这些是一个函数不可逆的仅有的可能原因。

可逆原理　如果一个函数既是一对一的又是映上的，则该函数是可逆的。

4.3　模算术函数

4.3.1　模加和加法逆元

图 4.6 描述的函数可以使用法则 $x \mapsto x+2 \text{ rem } 6$ 来描述。它的逆函数由图 4.7 描述，可以使用法则 $y \mapsto y+4 \text{ rem } 6$ 来描述。

后一个法则中 4 是做什么用的？在形式化的法则中 4 和 2 之间有什么关系？答案体现在以下同余式中：

$$2+4 \equiv 0 \pmod 6 \tag{4.1}$$

该同余式表明在模 6 加法中 4 扮演 -2 的角色。的确，-2 的模 6 代表元就是 4。鉴于同余式（4.1），我们说 4 是 2 的模 6 *加法逆元*（反之亦然）。

更一般地，如果 a 和 b 是满足以下同余式的整数：

$$a+b \equiv 0 \pmod m$$

我们说 a 和 b 互为模 m 的加法逆元。

考虑以下函数，其定义域是 $\{0,1,2,\cdots,m-1\}$，且由法则 $x \mapsto x+a \text{ rem } m$ 描述。如果 b 是 a 的模 m 加法逆元，则逆函数可以用法则 $y \mapsto y+b \text{ rem } m$ 描述（可以使用代换原理来证明）。

4.3.2　计算模 m 加法逆元

是不是每一个整数 a 都有模 m 加法逆元？如果是这样，我们如何找到它？在这里，考虑常规的运算是有帮助的。由于 $b=-a$ 满足等式 $a+b=0$，所以当然有 $a+b \equiv 0 \pmod m$。因此 $-a$ 是 a 的一个模 m 加法逆元。$-a$ 的模 m 代表元是 $m-a$。因此 $b=m-a$ 也是 a 的一个模 m 加法逆元（并且作为一个代表元更有优势）。 37

4.3.3　模乘和乘法逆元

乘法的情况类似，但是稍微复杂一些。考虑以下函数，其定义域是 $\{1,2,$

3,…,6} 且法则是 $x \mapsto x \cdot 2 \text{ rem } 7$。这个函数如图 4.9 所示。

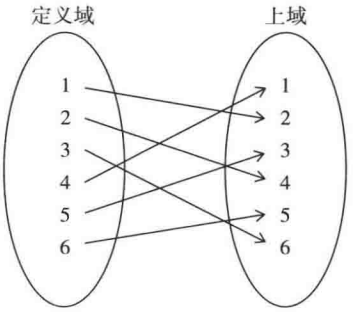
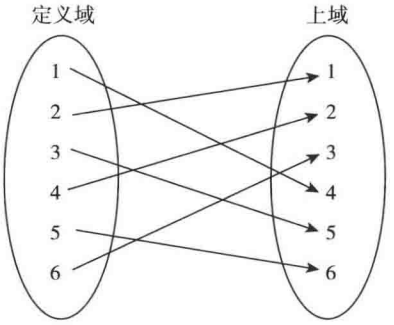


图 4.9 该函数的定义域是 {1,2,3,4,5,6}，法则是 $x \mapsto x \cdot 2 \text{ rem } 7$

这个函数将 1 映射到 2，2 映射到 4，3 映射到 6，以此类推。它的逆函数将 2 映射到 1，4 映射到 2，6 映射到 3，等等。逆函数如图 4.10 所示。



38

图 4.10 图 4.9 中函数的逆函数

这个逆函数可以用函数 $y \mapsto y \cdot 4 \text{ rem } 7$ 描述，这是下面同余式的一个推论：

$$2 \cdot 4 \equiv 1 \pmod 7 \tag{4.2}$$

该同余式表明，4 在模 7 乘法中扮演 1/2 的角色。鉴于同余式 (4.2)，我们说 4 是 2 的模 7 *乘法逆元*（反之亦然）。

更一般地，如果 a 和 b 是满足以下同余式的整数：

$$a \cdot b \equiv 1 \pmod m$$

我们说 a 和 b 互为模 m 的乘法逆元。

在这种情况下，由 $x \mapsto x \cdot a \text{ rem } m$ 和 $y \mapsto y \cdot b \text{ rem } m$ 描述的函数是相互为逆的。(这可以使用代换原理来证明。)

乘法逆元将在以后章节中要学到的许多密码学方案中扮演重要角色。

4.3.4　计算模 7 乘法逆元的简单方法

为了找到其他的模 7 乘法逆元，我们针对模 7 代表元写下模 7 乘法表：

	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	4	3	2	1	2
6	0	6	5	4	3	2	1

逐一搜索这个表，可以看到：

- 1 是它自己的乘法逆元（$1 \cdot 1 \equiv 1 \pmod 7$）；
- 2 和 4 互为乘法逆元（$2 \cdot 4 \equiv 1 \pmod 7$）；
- 3 和 5 互为乘法逆元（$3 \cdot 5 \equiv 1 \pmod 7$）；
- 6 是它自己的乘法逆元（$6 \cdot 6 \equiv 1 \pmod 7$）。

无疑这种找乘法逆元的方法可以对任意模 m 使用：只要对所有模 m 的代表元写下模 m 乘法表，然后逐一搜索即可。在第 8 章，我们会学习一种方法，当 m 很大的时候该方法会更加实用。

4.3.5　乘法逆元不总是存在

现在考虑以下函数，其定义域是 $\{1, 2, 3, 4, 5\}$ 且法则是 $x \mapsto x \cdot 2$。该函数如图 4.11 所示。注意到 $3 \cdot 2 = 0$，因此使用这一法则，我们必须把 0 加入到上域中。该函数是不可逆的。（它既不是一对一的，也不是映上的。）这反映了 2 没有模 6 乘法逆元的事实。

39

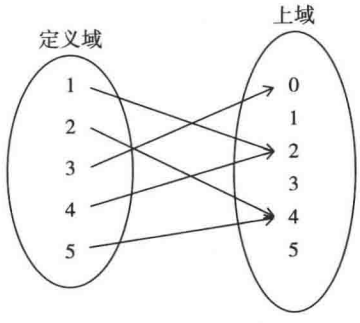


图 4.11　该函数的定义域为 $\{1, 2, 3, 4, 5\}$，法则是 $x \mapsto x \cdot 2 \text{ rem } 6$

为了验证这一点，我们写下针对模 6 代表元的模 6 乘法表：

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

逐一搜索这个表，我们发现 1 是其自己的乘法逆元，且 5 也是其自己的乘法逆元，但是没有其他代表元有乘法逆元。你或许会发现这有点令人惊讶。毕竟，在常规的运算中，对每一个非零数 a，存在一个数 b 满足 $a \cdot b=1$。（也就是，$b=1/a$。）在第 8 章，我们将研究什么时候一个整数才有模乘法逆元这个问题。

4.4 函数符号

数学家们经常使用名字来指代函数。“嗨，Bob!”不，实际上，数学家们很少使用“Bob”来指代一个函数。相反，他们使用“f”来表示函数。例如，一个数学家可能会说：“令 f 表示一个函数，其定义域是 $\{0,1,2,3,4,5\}$，且法则是 $x \mapsto x^2$，则 f 是一个一对一函数。”（他们使用稍微不同的术语，例如，“法则”不是很准确的数学用语。）给函数命名以后，我们可以表示与某一特定输入相关的输出。

40

为了表示与 3 相关的输出（3 的像），我们可以写成 $f(3)$。因此在这种情况下 $f(3)$ 的值是 9。由这样简明的符号来表示与某一特定输入相关的输出是非常有用的。

函数的逆函数也有符号。函数 f 的逆函数表示为 f^{-1}。例如，我们可以将 f^{-1} 与输入 9 相关的输出写成 $f^{-1}(9)$，在这种情况下是 3。因此在函数 f^{-1} 中 9 映射到 3。

当然，数学家们不可能使用相同的名字“f”来指代他们遇到的所有函数，对不对？然而，实际上，他们经常这么做。原因是 f 是某种临时的名字，它经常被用来指代与当前讨论相关的任何函数。因此，今天 f 可能指的是图 4.3 的函数，明天可能指的就是一个完全不同的函数，甚至是一个有不同定义域的函数。当然，好的数学表达需要清楚地表明所指的是哪一个函数。

当数学家们需要在同一个讨论中表示多个函数时，他们机智地求助于字母 g 和 h 等（这并不是数学创新能力最好的例子）。有时候，他们使用一个字母和数字下标的组合形式来表示函数。因此 f_1 可能表示一个函数，f_2 可能表示另一个不同的函数，f_3 是第三个，以此类推。使用这种方式几乎不可能用完名字。

这种下标方案在表示许多相似的函数时会起到很大的作用。例如，考虑“加 3”函数。有许多这种形式的函数：“加 1”函数，“加 2”函数，“加 4”函数，等等。一个数学家立刻可以通过以下方式定义所有这些函数。“对每个数 b，令 f_b 表示定义域为 $\{0, 1, 2, 3, 4, 5\}$ 且法则为 $x \mapsto x+b$ 的函数。”现在“加 311”函数就可以表示为 f_{311}。

4.5　函数的使用

函数提供了一种广泛应用的术语和符号，它们的灵活性的关键在于它们的抽象性。这里有一些非密码学例子（密码学例子以后会给出）。这些函数中哪些是一对一的？哪些是映上的？

- 定义域是布朗大学的本科生集合，法则是

$$x \mapsto x \text{ 的 ID 号}$$

41

- 定义域是布朗大学的本科生集合，法则是

$$x \mapsto x \text{ 头上的头发数目}$$

- 定义域是布朗大学的职员集合，法则是

$$x \mapsto x \text{ 的薪水}$$

- 定义域是所有活着的人，法则是

$$x \mapsto x \text{ 的母亲}$$

- 定义域是所有商业音响的集合，法则是

$$x \mapsto x \text{ 的制造商}$$

当然，我们可以很简单地针对一个不真正是函数的事物指定一个定义域和法则。以下哪些描述了函数？

- 定义域是布朗大学的学生集合，法则是

$$x \mapsto x \text{ 的室友}$$

- 定义域是布朗大学的学生集合，法则是

$$x \mapsto x \text{ 成为一个学生以来的学期数}$$

- 定义域是布朗大学的学生集合，法则是

$$x \mapsto x \text{ 到目前为止得到的最好的学期成绩}$$

4.6 一个两输入函数：一般化凯撒密码的加密函数

到目前为止我们考虑的函数都是单输入函数（通常被称为一元函数）。有一些概念（不是可逆函数!）很容易扩展到两输入函数。这里我们考虑一个很有用的例子，即凯撒密码的一个变体的加密函数。两个输入是（1）明文和（2）密钥；输出是密文。我们可以把它示意性地表示为

42

$$f(\textit{plaintext symbol}, \textit{key}) = \textit{cyphertext symbol}$$

两个输入中的每一个都应该是集合 $\{0,1,2,3,\cdots,25\}$ 中的元素，且输出也是该集合中的一个元素。该函数的法则是

$$f(\textit{plain}, \textit{key}) = \textit{plain} + \textit{key} \text{ rem } 26$$

其中我们已经使用 *plain* 和 *key* 来取代 x 和 y，目的是提醒我们自己记住这些输入的目的。

4.7 特殊化：将两输入函数转化为单输入函数

现在假设两个参与方 Alice 和 Bob 已经选择了一个密钥，如 17，用于加密他们的消息。我们可以定义一个新的函数 g 来表示使用该密钥的加密过程。函数 g 以 $\{0,1,2,3,\cdots,25\}$ 作为其定义域和上域。该函数使用以下法则定义：

$$g(\textit{plain}) = f(\textit{plain}, 17)$$

也就是说，我们根据 f 定义 g。g 的输入是明文，输出是密文。与之相对应的解密函数是什么？我们将 g 的逆函数写成 g^{-1}。因此 g 最好是可逆的。

如果不是引入一个新的名字 g 来表示带密钥 17 的加密函数，我们可能会使用下标，即 $f_{key=17}$ 来表示该加密函数。（相同的函数，用不同的方式命名它。）

通过将两输入函数中的某一个输入固定，从而得到一个单输入函数，这种方式有时候被称为特殊化。

如果对每个可能的密钥 k，$f_{key=k}(\textit{plain})$ 是一个可逆函数，我们说加密函数 $f(\textit{plain}, \textit{key})$ 满足唯一解密性。

注意到，我们可以使用另外一种方式特殊化函数 f，即选择一个明文符号（比如使用 2 来表示“c”）而不是一个密钥。也就是说，我们可以使用以下法则来定义函数 h：

$$h(key)=f(2,key)$$

函数 h 的输入是一个密钥，输出是使用该密钥对“c”的加密值。该函数的用处显然不如函数 g，但是我们稍后会看到它为什么在评估一个密码系统的安全性时是有用的。

让我们来看一个特殊化的简单例子。我们从一个描述两输入函数的表格开始：

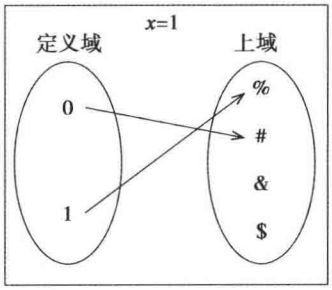
43

x \ y	1	0
1	%	#
0	&	$

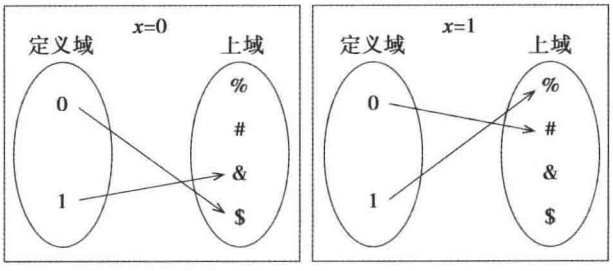
现在固定 x 的输入为 1。我们现在只看一下表格中的相关行：

x=1 \ y	1	0
	%	#

这可以理解为一个带有单输入 y 的函数。表示如下：



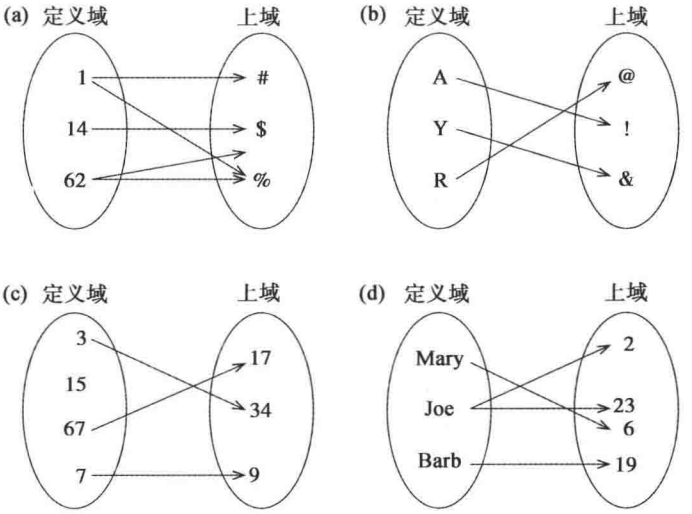
类似地，可以固定输入 x 为 0。下面是我们所得到的函数：



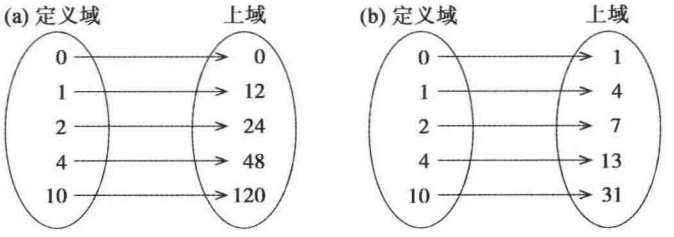
4.8 思考题

1. 凯撒密码的加密函数看起来像 $f(plain, key) = (plain + key) \text{ rem } 26$。令
44 $g(cyph, key)$ 为解密函数。也就是说，针对一个密文 $cyph$，$g(cyph, key)$ 是相对应的明文符号。给出函数 g 的法则。($g(cyph, key) = \cdots ?$)

2. 针对以下每个图示，说明其是否是一个函数。



3. 针对以下每个图示，给出其相关的法则。

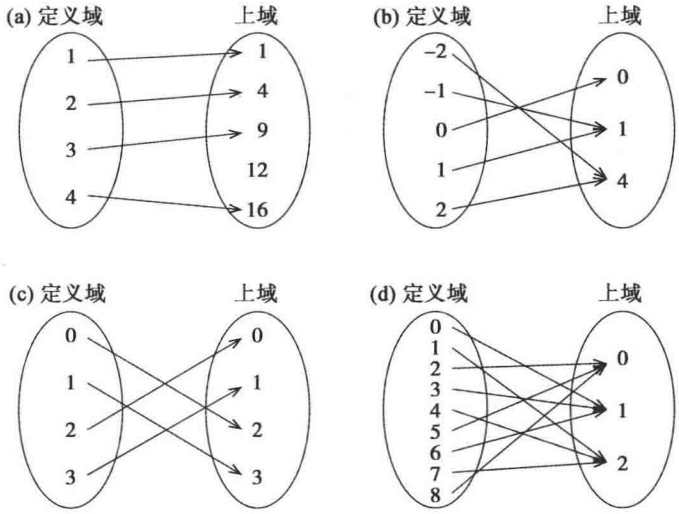


4. 考虑问题 3 中描述的函数。

(a) 给出函数 3(a) 的逆函数的法则。

45 (b) 给出函数 3(b) 的逆函数的法则。

5. 分别针对以下图示中所描述的函数，给出一个法则：



6. 考虑问题 5 中的图示。针对每一个函数，如果逆函数存在，给出其法则；如果逆函数不存在，解释为什么。(一对一和映上的概念在这里可能会有用。)

7. 考虑以下函数，其定义域和上域为 {0，1，2，3，4，5，6，7，8}。针对以下每一个法则，给出其逆函数的法则。

   (a) $x \mapsto x+1 \text{ rem } 9$

   (b) $x \mapsto x+2 \text{ rem } 9$

   (c) $x \mapsto x+3 \text{ rem } 9$

   (d) $x \mapsto x+6 \text{ rem } 9$

   (e) $x \mapsto x+0 \text{ rem } 9$

8. 准备两个模乘法表，一个的模数是 11，另一个的模数是 15。使用你的表格，寻找以下指定元素的乘法逆元。对其中每个元素，如果没有乘法逆元，验证你的答案。

   (a) 5 的模 11 乘法逆元

   (b) 4 的模 15 乘法逆元

   (c) 5 的模 15 乘法逆元

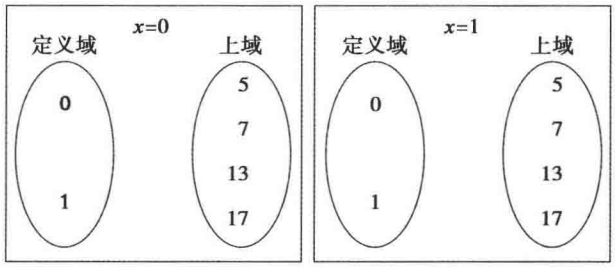
   (d) 11 的模 11 乘法逆元

   (e) 3 的模 15 乘法逆元

9. 下面我们使用一个表格来描述一个带有两输入 x 和 y 的函数。你的工作是依次对 x 取一个值，将那个值代入，并获得一个单输入函数。

		y	
		1	0
x	1	7	13
	0	17	5

填写以下图示中的箭头，以此得到适当的单输入函数图。

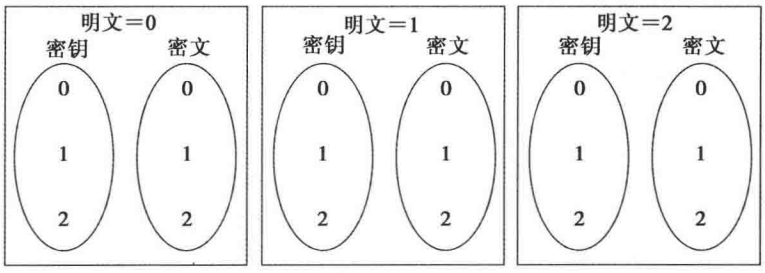


10. 我们已经讨论了一个密码系统如何被描述成一个两输入函数。也就是说，$f(plain,key)=cyph$。正如我们在第 4 章中所言，可以固定或者密钥或者明文输入，并针对该固定输入的所有可能值，构造单输入的函数。

(a) 考虑以下两输入函数表，其描述了一个密码函数，该函数加密的字符来源于一个只有三个字符的字母表。也就是说，明文和密文都是 0，1 或 2。

		key	
		0	1
	0	0	1
plain	1	1	2
	2	2	0

将明文输入作为固定变量，并填写所提供的单输入函数图，这样就得到该函数的一个完整描述。



(b) 想象一下你是 Eve，你看到了一个由 Alice 传送的使用上面描述的加密

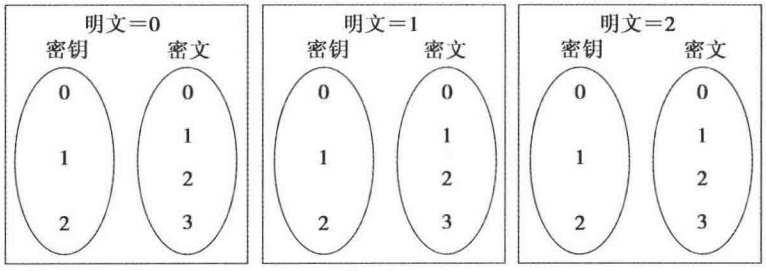
函数加密的密文。该密文是 2。给定该信息，其中有一个明文是不可能的。是哪一个？

11. 现在我们将使用一个法则来描述一个加密方案，并像在上一个题目中一样做相同的分析。为了使事情不复杂，我们将使用以下方案，其被设计用来加密从一个非常小的字母表中选择的字符。再一次想象一下我们将加密以下三个可能明文：0，1 或 2。就像凯撒密码，我们将从 0，1 或 2 中选择一个密钥，并把它加到明文中去。然而，在这系统中，我们将做模 4 加法。因此，$f(plain,key)=plain+key \pmod 4$。

(a) 针对这一加密函数制作一个两输入函数表格。

		key		
		0	1	2
plain	0			
	1			
	2			

(b) 使用与上面相同的方法，将该函数描述为一个单输入函数的集合，其中对每一个函数，其明文输入被固定。



(c) 现在想象一下你是 Eve，而且你注意到了一个密文 3，它可能是由哪个明文加密得来？

48

第5章
A Cryptography Primer: Secrets and Promises

概　率　论

5.1　实验结果

考虑一个依赖于偶然性的实验，这个实验不依赖于任何未知的因素。关于实验结果，我们能说什么？如果我们真正进行了这个实验，则能准确地说出发生了什么，但是在实验进行之前，我们能说什么？我们希望能给出的最好描述是每一个可能结果的可能性。这样的描述被称为概率分布。概率论就是关于可能性的推理方法。

考虑掷一粒骰子，有6个可能的结果（不包括像“骰子滚落桌下”等古怪的事情）。我们隐式地假设一次实验的结果是相互排斥的，也就是每次执行实验只能产生一个结果。

可能的结果如图5.1所示。

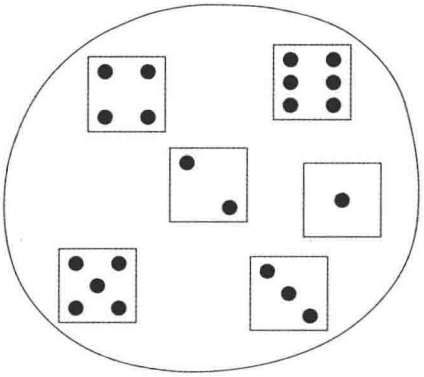


图5.1　掷一个骰子的概率空间

可能结果的集合被称为样本空间，也被称为概率空间。

5.2　结果的概率

我们为6个实验结果中的每一个实验结果指定一个数字表示它的概率，由

此来描述相关结果的可能性。例如，如果一个实验结果发生的可能性是另一个结果的两倍，我们指定第一个结果的概率是第二个结果概率的两倍。

某些约定支配着我们用作概率的数字。一个实验结果的可能性是另一个实验结果可能性的－1 倍是没有意义的，所以要求概率是一个非负数字。我们用概率为 0 来对应不可能发生的事件，即一个永远不会发生的实验结果的概率被赋值为 0。我们用概率为 1 来对应必然事件，即一个总是发生的实验结果的概率被赋值为 1。在一个典型的实验中，每个实验结果的概率是一个大于 0 小于 1 的数字。

这里有一个解释我们实验结果概率的方法。如果你准确地执行同一个实验数千次，则某个可能实验结果的概率是关于这个实验结果出现次数的比例的最好估计。例如，假设我们掷一粒骰子 1000 次，对掷出 2 点的次数的最好估计是 1000 的 1/6。在这个实验中，掷出 2 点的概率是 1/6，可以写作

$$\text{Prob}[\boxed{\cdot\ \ \cdot}] = 1/6$$

类似地，对其他的实验结果：

$$\text{Prob}[\boxed{\cdot}] = 1/6,\ \text{Prob}[\boxed{\cdot\ \ \cdot}] = 1/6,\ \text{Prob}[\boxed{\cdot\ \ \cdot}] = 1/6,$$

$$\text{Prob}[\boxed{\cdot\ \ \cdot}] = 1/6,\ \text{Prob}[\boxed{\cdot\ \ \cdot}] = 1/6$$

形式化地，实验结果的概率由一个函数（不妨叫作 p）表示，它的定义域是样本空间，上域是 0 与 1 之间所有数字的集合。

5.3 绘制概率分布图

使用条状图来总结不同实验结果的概率通常是有用的。沿着横坐标列出所有可能的实验结果，而纵坐标标明了概率值。

49
〜
50

一个骰子上点数的概率分布如图 5.2 所示。

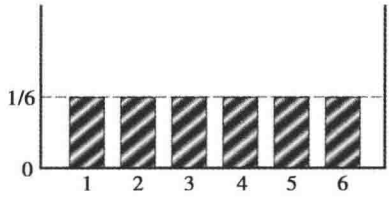


图 5.2 掷一粒骰子的概率分布

5.4 实验结果集合的概率

得到一个偶数的概率是什么？基于概率论，为了找到某几个实验结果发生的概率，可以把这些实验结果的概率加起来。因此得到⚁、⚂或⚃的概率是1/6+1/6+1/6。

我们可以通过类似的推导得到⚀、⚁、⚂、⚃、⚄或⚅的概率是

$$\frac{1}{6}+\frac{1}{6}+\frac{1}{6}+\frac{1}{6}+\frac{1}{6}+\frac{1}{6}$$

也就是1。这是可以预料的，因为6个面之一向上是必然发生的（没有其他可能的实验结果）。一般地，如果把一个实验所有可能结果的概率相加，你最好得到1。

5.5 小结

- 实验可能的实验结果是*互斥的*（不可能同时得到两个结果）。
- 可能的实验结果的列表必须是*详尽无遗的*。当进行实验时，列表中的结果必有一个出现[⊖]。

51

- 每个可能的实验结果有一个*概率*，它是一个0与1之间的数。
- 对任意可能实验结果的集合，可以计算这个集合成员发生的概率：将每个单独实验结果的概率相加。
- 所有实验结果的概率之和必须等于1。

5.6 均匀分布

可以使用最后一个事实来决定某个实验的概率。假设有k个实验结果，所有的实验结果都是等可能的（就像在掷骰子的情况下，$k=6$）。应该如何指派概率值呢？我们必须为每个实验结果赋相同的概率值，不妨用变量x表示这个未知的概率值。因为所有这些实验结果的概率之和必须等于1，于是有

$$\overbrace{x+x+\cdots x}^{k\text{个}}=kx=1$$

⊖ 有时这是一个真实实验在数学上的理想化。在一个现实掷币实验中，骰子滚落桌子（或者硬币立起来，不管什么）是可能发生的，但是在概率空间里，通常不把这些可能性作为实验结果。

也就是，$kx=1$。解出 x，我们得到 $x=1/k$。因此每个实验结果的概率是 $1/k$。

一个所有实验结果概率相等的概率分布称为均匀概率分布（经常只称之为均匀分布）。我们考虑另一个实验：掷两粒骰子。

有 36 种可能的实验结果（图 5.3），所有结果都是等可能的。因此每个可能实验结果的概率都是 1/36。这是另一个均匀概率分布的例子。

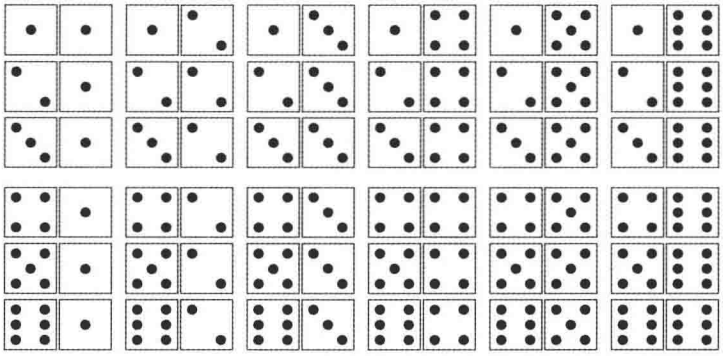


图 5.3

52

5.7 随机变量

我们经常要从一个实验的实验结果进行大量的推导。例如，在掷两粒骰子的情况下，我们经常对掷出骰子的点数总和感兴趣。我们使用随机变量的概念。可以把随机变量想成一个变量，如 X，它的值依赖于一个实验的实验结果。在两粒骰子的情况下，例如，可以说："令 X 表示掷两粒骰子之后的点数之和。X 等于 5 的概率是多少？"我们可以按如下所示，通过实验可能的结果来计算这个概率。（注意我们使用"Prob $[X=5]$"作为"X 等于 5 的概率"的数学表达式。）

$$\text{Prob}[X=5]=\text{Prob}[\boxed{\cdot}\ \boxed{\cdot\cdot}\ \text{或者}\ \boxed{\cdot\ \cdot}\ \boxed{\cdot\ \cdot}\ \text{或者}\ \boxed{\cdot\ \cdot}\ \boxed{\cdot\ \cdot}\ \text{或者}\ \boxed{\cdot\cdot}\ \boxed{\cdot}]$$

$$=\text{Prob}[\boxed{\cdot}\ \boxed{\cdot\cdot}]+\text{Prob}[\boxed{\cdot\ \cdot}\ \boxed{\cdot\ \cdot}]+\text{Prob}[\boxed{\cdot\ \cdot}\ \boxed{\cdot\ \cdot}]+\text{Prob}[\boxed{\cdot\cdot}\ \boxed{\cdot}]$$

$$=\frac{1}{36}+\frac{1}{36}+\frac{1}{36}+\frac{1}{36}=\frac{4}{36}$$

这里有另外一个计算。X 至少为 10 的概率是多少？

$$\text{Prob}[X\geqslant 10]=\text{Prob}[X=10\ \text{或者}\ X=11\ \text{或者}\ X=12]$$

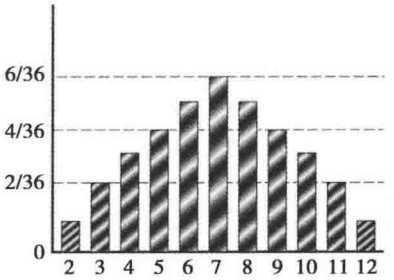
$$=\text{Prob}[X=10]+\text{Prob}[X=11]+\text{Prob}[X=12]$$

$$= \text{Prob}[\boxed{\cdot\,\cdot}\ \boxed{\cdot\,\cdot}\text{或者}\boxed{\cdot\,\cdot}\ \boxed{\cdot\,\cdot}\text{或者}\boxed{\cdot\,\cdot}\ \boxed{\cdot\,\cdot}]$$

$$+ \text{Prob}[\boxed{\cdot\,\cdot}\ \boxed{\cdot\,\cdot}\text{或者}\boxed{\cdot\,\cdot}\ \boxed{\cdot\,\cdot}] + \text{Prob}[\boxed{\cdot\,\cdot}\ \boxed{\cdot\,\cdot}]$$

$$= \frac{3}{36} + \frac{2}{36} + \frac{1}{36} = \frac{6}{36}$$

X 的概率分布如图 5.4 所示。



53

图 5.4 X 的概率分布

5.7.1 基于另一个随机变量定义随机变量

我们可以按照之前定义的随机变量来定义一个新的随机变量。令 $Y=(X-7)^2$。这里有一个关于 X 和 Y 的值的表格。

X	Y
2	25
3	16
4	9
5	4
6	1
7	0
8	1
9	4
10	9
11	16
12	25

注意到，例如 X 有两个值 2 和 12，对应的 Y 值都是 25。因此 Y 等于 25 的概率是 X 等于 2 或 12 的概率。由于 Prob[$X=2$] 等于 1/36，Prob[$X=12$] 也等于 1/36，可以得出 Prob[$Y=25$] 等于 1/36+1/36。

Y 的概率分布如图 5.5 所示。

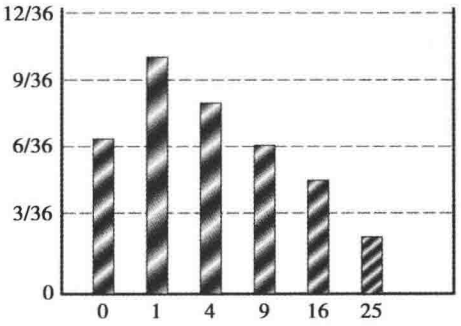


图 5.5 Y 的概率分布

54

5.7.2 随机变量的形式化数学定义

形式化地，一个随机变量既不随机也不是一个变量（你们自己讨论），它是一个定义域为样本空间的函数。

因此，随机变量 X（掷两粒骰子之后的点数之和）事实上是图 5.6 刻画的一个函数[⊖]。

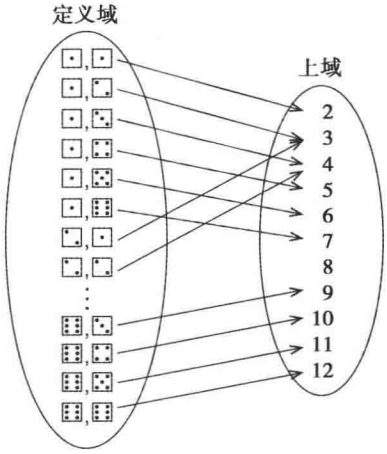


图 5.6 随机变量 X，掷两粒骰子之后的点数之和的函数

基于函数的推理可被用于基于随机变量的推理，反之亦然。我们使用这类推理与加密函数相关联。

⊖ 我们不真正关心这个函数的上域是什么，在这个图中只画出了值域，也就是能够成为这个函数的像的那些元素的集合。然而，在 5.7.3 节中，我们将看到为什么有时上域也有影响的原因。

5.7.3 随机变量的均匀分布

我们回到掷一粒骰子的实验。令 Z 是掷过之后骰子点数模 3 的代表元，则 Z 是一个随机变量，且 Prob[$Z=0$]=1/3，Prob[$Z=1$]=1/3，Prob[$Z=2$]=1/3。Z 的概率分布是不是一个均匀分布？好，这是有条件的，可以用不同的方法来描述
55 它（图 5.7）。

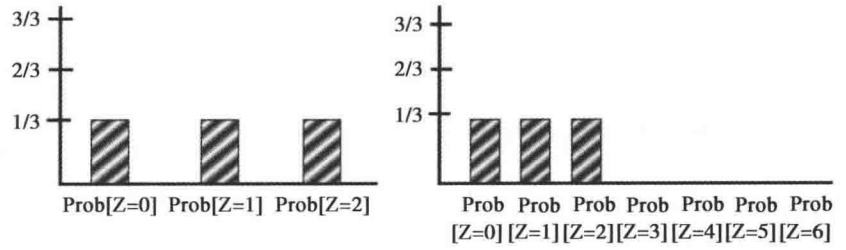


图 5.7 描述随机变量 Z 的不同方法

我们必须清楚，一个随机变量是否被认为是一个均匀分布依赖于我们打算考虑其概率的值。也就是说，人们可以通过加上一些概率为 0 的值（随机变量不可能取那些值）来声明概率分布不是均匀的（因为还有些其他值具有非零的概率）。因此，在决定一个随机变量是否是均匀分布时必须多加一点小心。我们将用两种方法来处理这些情况。

首先，记住一个随机变量是一个定义域为样本空间的函数。作为一个函数，它有一个上域。（同样回忆上域中可能存在一些值不在值域中，这些值发生的概率为 0。）为了描述随机变量的概率分布，在某些场合中，考虑上域中所有元素的概率是有理由的，因此我们可以说，当所有上域中元素的概率均相同时，随机变量是均匀的。

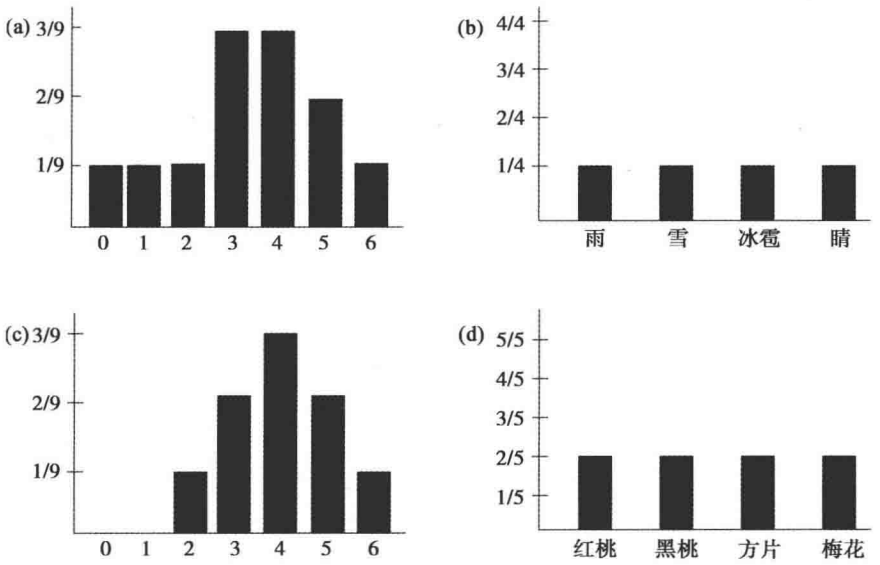
一个更安全的解决方案是：当讨论一个随机变量的分布是否均匀的时候，我们必须精确地指明哪个值的集合是被考虑的。为此，我们说“在 S 上均匀分布”来代替“均匀分布”，其中 S 是被考虑的值的集合。

通常，我们要清楚将术语“均匀”用于随机变量的概率分布有一点模糊。这些问题在分析密码系统的安全性时将会出现。

5.8 思考题

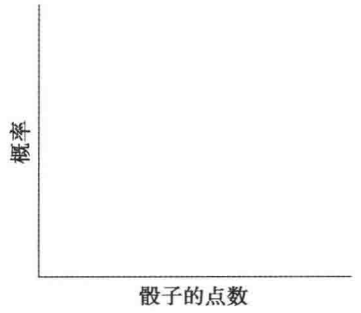
1. 判断下面的每一个图是否表示一个正确的概率分布，如果是，指出它是均匀分

布还是非均匀分布。 56



2. 考虑一粒 5 面的骰子。

(a) 掷这种骰子的可能值的集合是从 1 到 5 的整数。画出这个可能值的集合的概率分布图，并在坐标轴上标出这些值。 57

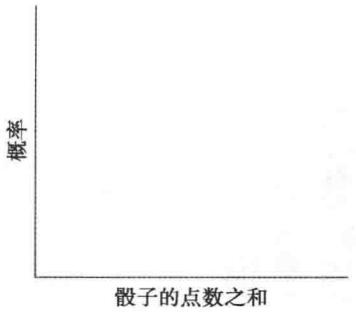


(b) 这是一个在这些可能值集合上的均匀分布吗?

3. 现在考虑两粒 5 面的骰子。

(a) 如果你掷了这两粒骰子，那么它们的点数之和的可能值是什么?

(b) 画出这个可能值的集合的概率分布图，并在坐标轴上标出这些值。不要在图上放入任何概率为 0 的值。



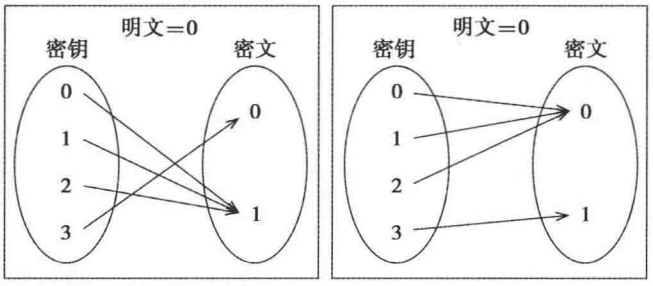
(c) 这是一个在实验 (a) 中的可能值集合上的均匀分布吗?

4. 现在考虑下列加密函数表。这个加密函数被设计用于加密 1 或者 0。

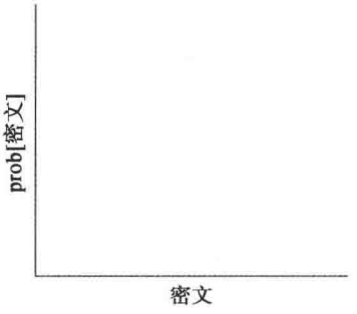
明文 \ 密钥	0	1	2	3
0	1	1	1	0
1	0	0	0	1

58

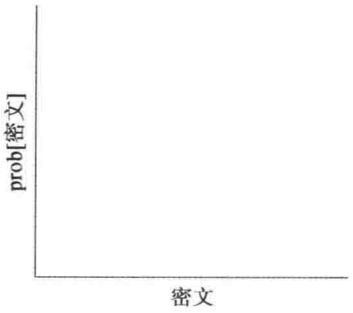
通过对明文特殊化得到对应的单输入函数如下:



(a) 想象一个实验,在这个实验中,密钥是被随机均匀选择的。令随机变量 A 是对明文 0 的加密,画出 A 的概率分布图。



(b) 令随机变量 B 是对明文 1 的加密,画出 B 的概率分布图。

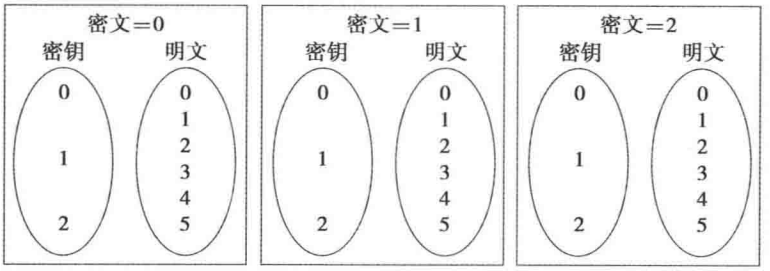


59

5. 考虑下列解密函数表：

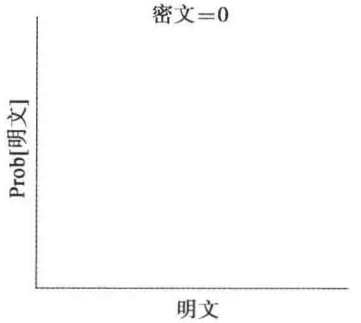
明文 \ 密钥	0	1	2
0	2	1	0
1	0	1	2
2	1	0	2
3	2	0	1
4	1	2	0
5	0	2	1

(a) 在下列函数图上填入箭头，这些函数图表示了对上述解密函数的特殊化：



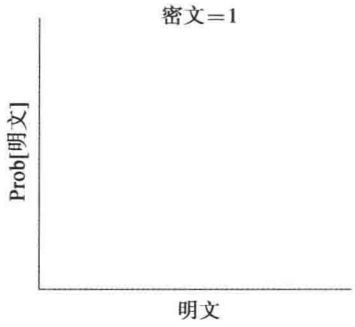
(b) 令密钥是在集合 {0，1，2} 上按照均匀分布随机选取的。

i. 令 A 是使用随机密钥对密文 0 的解密，画出 A 的概率分布图。

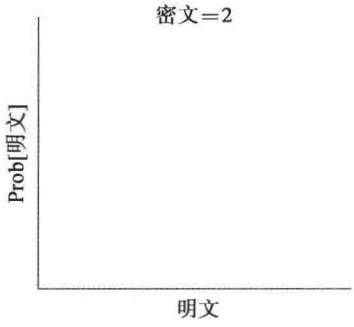


60

ii. 令 B 是对密文 1 的解密，画出 B 的概率分布图。



iii. 令 C 是对密文 2 的解密，画出 C 的概率分布图。



61

第6章
A Cryptography Primer: Secrets and Promises

完美保密与完美安全的密码系统

数学家克劳德·香农（Claude Shannon）第一次形式化地定义了完美保密的概念，并展示了一个特定的密码系统以实现完美保密性。我们没有涉及香农全部的理论知识，而是重点关注与密码学研究直接相关的一部分，这一部分所依赖的数学知识在本门课程范围之内。

在这一章我们讨论怎么判断一个密码系统是否满足完美保密性。在第7章我们讨论使用完美保密的更多方法。通过这些内容，读者将更加清楚：为什么大多数密码体制更多地使用模运算，而不是传统的算术运算。

我们将看到，一个密码系统的完美保密性和唯一解密性在数学上是一对表亲关系。一个密码系统可能是唯一解密的，但是不满足完美保密性，反之亦然。然而，决定一个密码系统是否满足完美保密性与决定它是否是唯一可解密的数学原理极其相似。

6.1 窃听者能够从密文中获得什么

一个密码系统是完美保密的，如果窃听者从密文中得不到关于明文的任何信息。因此，为了理解什么样的密码系统是安全的，我们要考虑“得到某些信息”是什么含义。

为此，我们考虑一个非常简单的场景。Alice向Bob发送一条加密后的消息，Eve窃听到它（目前，我们忽略Bob使用相同的密码系统甚至相同的密钥回复的可能性）。为了了解Eve从密文中获得了什么，我们考虑她在得到密文之前所了解到的明文信息（她的*先验*知识）和得到密文之后所了解到的明文信息（她的*后验*知识）。因为我们不希望密码系统的安全性取决于Eve拥有什么样的知识，所以想要建立一个关于“Eve了解到什么”的一般模型。假设Eve拥有一个Alice可能发送的所有明文的列表（可能是一个隐含的列表）。另外，对于每个可能的明文，Eve分配一个正数来表示这个可能的明文是真正明文的可能性有多大。换言之，Eve知道真实明文的*概率分布*。

当然，在现实中，明文的选择通常不是随机的。Allice 通常有一个非常特别的消息想要发送给 Bob。然而，我们现在从 Eve 的视角研究这个场景。对于她来说，明文是不可预测的，就像它是被随机挑选的一样。因此就数学而言，Eve 认为 Alice 按照某个特定的概率分布随机挑选明文。

原则上，Eve 关于明文的先验知识可以用任何概率分布来表示。然而，为了简单起见，我们通常假设它是在某个子集上的均匀分布。

例 1 在看到密文之前，Eve 知道明文是一个美国的电话号码，因此由 10 位数字组成。在这种情况下，她的先验知识可能是对每个 10 位的数字分配相同的概率。

如果 Eve 还有一点小聪明，她可能会有关于明文更精确的模型。例如，她可以只考虑那些前三位数字可以形成有效区号的 10 位数。在这种情况下，她的（明文）概率分布是在这个数字集合上的均匀分布：所有这种 10 位数被赋予相同的概率，其他数的概率被赋值为 0。 ■

例 2 有一个历史性例子，我们考虑它的变体。Eve 可能提前知道明文要么是 0 要么是 1（0 表示英国计划在陆地上发动攻击，1 表示从海洋发动攻击）。基于她对英国军事策略的了解，Eve 可能会认为在陆地上发动攻击更有可能。在这种情况下，她可能赋予陆地攻击 2/3 的概率，海洋攻击的概率为 1/3。 ■

Eve 关于明文的后验知识呢？也就是说，看到密文之后，她了解到什么？记住我们假设 Eve 知道 Alice 和 Bob 使用的密码系统（但她不知道密钥）。Eve 知道存在某个密钥可以解密密文产生明文。有时这可以提供一些关于明文的额
63 外信息。

例 1a 假设正在使用的密码系统是加法密码，其中分组长度是五位数。在这种情况下，模数是 10^5，密钥是五位数字。假设截获到的两个密文分组是 37491……92722。Eve 知道每个美国区号的第二位数字是 0 或 1（这一般是正确的）。通过检查第一个密码分组的第二位数，她可以推断出密钥的第二位数是 5，6 或 7。然后再检查第二个密文分组的第二位数，她可以推断出第二个明文分组的第二位数是 4，5，6 或 7。在获得密文之前，她不知道这些信息，因此她至少可以从密文中得到一点点明文信息。 ■

现在我们考虑任意的密码系统。令 $g(cyphertext, key)$ 是解密函数，假设 b

是系统所允许的最大密钥（也就是，密钥是 0 到 b 之间的任意数）。假设 Eve 获得了密文，用字母 c 表示密文。那么她可以推断出明文是下面中的一个：

$$g(c,0),g(c,1),g(c,2),\cdots,g(c,b)$$

她可以将这些信息与先验知识结合起来，从而进一步缩小可能的答案集合。

例 3 假设可能的密钥有 10^{30} 个。在这个例子中，我们不指明密码系统。假设 Eve 知道明文是一个长度为 75 的英文句子。大概有 2^{75} 个这种句子。她的关于明文的先验知识用一个概率分布来表示，这个分布是在所有这种句子上的均匀分布。

现在她知道密文是 7856…03457（某个特定的 150 位数）。有多少对明文的猜测与她的先验知识和密文知识一致？答案取决于密码系统和特定的密文，但是一个好的猜想是非常少的。在对密文的所有可能的解密中，只有 1 个或 2 个或 10 个或很少个是英文文本。想象一下 Eve 试了所有可能的密钥，并用以解密密文，丢弃不是英文文本的结果，结束后，她只保留了很少的可能结果。如此一来，在这个例子中，她可以从密文中获得大量关于明文的信息。 ■ 64

一个复杂的公式可以将 Eve 关于明文的后验知识表示成一个概率，这个概率与 Eve 的先验知识和她截获的密文有关。这个公式使用了贝叶斯定理来帮助进行收集知识的概率计算。我们不会探究这个公式。对于我们来说，一个重要的特殊情形是 Eve 关于明文的先验知识用在某个集合 S 上的均匀分布来表示。在这种情况下，Eve 获得密文后，其关于明文的后验知识是在集合 S 与密文所有可能解密结果集合（也就是通过实验所有可能密钥得到的集合 $g(c,0),g(c,l),\cdots$）的交集上的均匀分布。

6.2 密码系统的评估

从 Eve 的视角研究过问题之后，现在我们转向一个虚构的组织的视角看待问题。这个被称作 NSO（No Such Organization）的虚拟组织正在推荐采用的密码系统。NSO 试图保证：如果密码系统被正确使用，它将是完美保密的，窃听者 Eve 从密文中得不到任何信息。

假设我们正在评估一个密码系统，其可能的密钥是 0 到 s 之间的整数，可能的明文集合是 0 到 t 之间的整数，加密函数是 $f(clear,key)$。

NSO指定密钥是在所有可能密钥集合上均匀随机选取的。为了分析，将密钥表示为一个随机变量 K。使用密钥加密明文得到密文，因此，密文取决于明文和密钥。对于每个可能的明文 i，我们用 X_i 表示相应的密文。也就是说，$X_i = f(i, K)$。请注意，因为密钥 K 是一个随机变量，所以 X_i 也是一个随机变量。

下面是完美保密的安全性要求：

完美保密 随机变量 X_0，X_1，X_2，…，X_t 的概率分布是相同的。

因此，如果有人只告诉你 X_i 中的一个随机变量的概率分布，你不能判断这是哪一个，也就是说，是哪个明文产生了它。如果她不告诉你概率分布，而是
65 给你一个根据此分布选择的数，你还是不知道使用了哪一个 X_i。

但是这恰恰是Eve得到的：随机变量 X_i 中的一个特定值。这个特定值是根据变量的概率分布随机选择的。因为所有变量的概率分布是相同的，Eve得不到关于特定明文 i 的任何信息。她的*后验*知识与她的*先验*知识相同。

例 2a 考虑一个密码系统，其加密函数是 $f(clear, key) = clear + key \text{ rem } 2$。明文仅可能是1和0，且密钥也仅可能是1和0。密钥是随机选取的。密钥有1/2的概率是0，有1/2的概率是1。

- 假设明文是1，那么密文的概率分布是什么呢？密钥有1/2的概率是1，所以密文是 $1+1 \equiv 0$。密钥有1/2的概率是0，所以密文是 $1+0 \equiv 1$。因此密文有1/2的概率是0，有1/2的概率是1。
- 假设明文是0，那么在这种情况下密文的概率分布是什么呢？密钥有1/2的概率是1，所以密文是 $0+1 \equiv 1$。密钥有1/2的概率是0，所以密文是 $0+0 \equiv 0$。因此密文有1/2的概率是1，有1/2的概率是0。 ■

我们看到这个密码系统实现了完美保密性：对于均匀随机选择的密钥，密文的概率分布不依赖于明文。

例 4 考虑一个基于非模算术的密码系统，其加密算法是 $f(clear, key) = clear + key$。假设明文和密钥必须是模26的代表元。因此，密文可以是0到50之间的任意数。我们假设密钥是在 {0，1，2，…，25} 上均匀随机选择的。

- 假设明文是0。在这种情况下，密文在 {0,1,…,25} 上是均匀分布的。密文是26或是27……或是50的概率是0。

- 假设明文是 1。在这种情况下，密文均匀分布在 {1,2,…,26} 上。
- 假设明文是 25。在这种情况下，密文均匀分布在 {25,26,…,50} 上。 ■ 66

很明显，密文的概率分布严重依赖于特定的明文。因此这个系统没有达到完美保密。事实上，举例来说，一个具有先验知识明文是 0 或 25 的窃听者很有可能在获得密文后就能知道到底是哪一个明文。因为只存在一个密文，也就是 25，对应的明文既可以是 0，又可以是 25。

即使是 NSO 也不能期望单独地考虑每个明文 i，并计算其对应密文随机变量 X_i 的概率分布。相反，研究者们利用加密函数的数学性质来说明完美保密性被达到。

他们使用了下面的引理。（引理是为了帮助证明其他结论而被证明的数学结论。）下面这个引理的证明可能看上去难以理解，但是思想很简单。

引理 1 *假设 $g(x)$ 是一个可逆函数。令 X 是 g 的定义域上的一个元素，X 是按均匀分布随机挑选的（也就是说，X 是一个随机变量），那么相应的输出 $g(X)$ 是随机的，并且均匀分布于上域的元素里。*

证明 假设 c 是上域中的任意一个元素，根据 g 的逆函数 g^{-1} 的定义，随机上域元素 $g(X)$ 等于 c，只要随机定义域元素 X 等于 $g^{-1}(c)$。$X=g^{-1}(c)$ 的概率是 1 除以定义域的元素数，所以 $g(X)=c$ 的概率也是 1 除以定义域的元素数。

我们已经证明，给定上域中的任意一个元素，$g(X)$ 等于这个元素的概率是 1 除以定义域的元素数。因为所有上域中的元素具有相同的概率，所以 $g(X)$ 的分布是均匀的。 ■

例 5 考虑函数 $g(x)=x+3 \text{ rem } 4$，其定义域为 {0，1，2，3}，上域也是 {0，1，2，3}。假设 X 是均匀随机选取的定义域中的元素。这意味着，X 有 1/4 的概率是 0，1/4 的概率是 1，1/4 的概率是 2，1/4 的概率是 3。令 $Y=f(X)$，那么 Y 也是一个随机变量。$Y=0$ 的概率是多少？$Y=0$ 意味着 $X=1$，且以 1/4 的概率发生。$Y=1$ 的概率是多少？$Y=1$ 意味着 $X=2$，这个发生的概率是 1/4。同样，$Y=2$ 的概率是 1/4，$Y=3$ 的概率也是 1/4。因此，Y 在 {0，1，2，3} 上是均匀分布的。 ■ 67

引理 2 *对于每个明文 i，定义函数 $f_{dear=i}(key)$ 为*

$$f_{dear=i}(key) = f(i,key)$$

如果对于每个明文 i，函数 f_i 是可逆的，那么使用加密函数 f 的密码系统是完美保密的。

证明 假设密钥 K 是根据均匀分布随机选取的。令 i 表示任意一个明文。密文是随机变量 $f_i(K)$。由引理 1 得，这个随机变量的概率分布是均匀的。

我们已经证明，不管选取什么明文 i，密文的概率分布是均匀的。因此，密文的概率分布不依赖于我们选取的明文。这表明这个密码系统实现了完美保密。 ■

例 5a 考虑模 4 加法密码，加密函数是 $f(clear, key) = clear + key \text{ rem } 4$。假设密钥是在 {0，1，2，3} 上均匀随机选取的。

- 假设明文 i 是 0，通过将公式中的明文赋值为 0，我们得到函数

  $$f_0(key) = 0 + key \text{ rem } 4$$

  这当然是一个可逆函数（事实上，这个函数的逆是它本身）。因此，由引理 1 得，对于一个均匀随机的密钥，密文在 {0，1，2，3} 上是均匀随机的。
- 假设明文 i 是 1，通过将公式中的明文赋值为 1，我们得到函数

  $$f_1(key) = 1 + key \text{ rem } 4$$

  因为这是一个可逆函数，所以密文在 {0，1，2，3} 上是均匀随机的。
- 可以试验明文等于 2 和明文等于 3 的情况。在这些情况下，密文在 {0，1，2，3} 上也是均匀随机的。

68

不管明文是什么，相应密文的概率分布在 {0，1，2，3} 上是均匀的。因此这个密码系统实现了完美保密。 ■

例 1b 我们再一次考虑分组长度是五位数的加法密码，模数是 10^5，密钥是一个五位数字。然而相对于加密两个分组，这次我们考虑只加密一个分组的情况（如果想要安全地加密第二个分组，必须使用另一个密钥）。加密函数是

$$f(clear, key) = clear + key \text{ rem } 10^5$$

令 i 表示任意一个明文（也就是任意一个五位数）。那么之前定义的函数 f_i 是

$$f_i(key) = i + key \text{ rem } 10^5$$

因此这个函数将 i 模加到它的输入之上。这个函数可逆吗？是的：模加 i 的

逆是模减 i。因此逆函数 g_i 以下面的法则定义：

$$g_i(key) = key - i \text{ rem } 10^5$$

因此，由引理 2 得，这个密码系统实现了完美保密。 ■

例 1c 现在假设我们使用与例 1b 相同的密码系统，但用同一个密钥加密两个明文分组。例如，假设明文是 40186 37600。密文也由两个分组构成。密文的第一个分组通过将密钥加到 40186 上（并取余数）得到。密文的第二个分组通过将密钥加到 37600 上（并取余数）得到。我们可以将这个函数的法则书写如下：

$$h(key) = [40186 + i \text{ rem } 10^5, 37600 + i \text{ rem } 10^5]$$

这个函数可逆吗？不，它不可逆。例如，密文 [40186，40186] 在函数 h 下不是任何密钥的像。它的逆函数不能将 40186 40186 映射到任何数。因此它的逆不是一个函数。 69

事实上，正如我们在例 1a 中看到的，这个密码系统没有实现完美保密性：Eve 可以从密文中获得关于明文的信息。利用概率分布的术语理解这一点的一个方法是，如果明文是 40186 37600，那么密钥有 $1/10^5$ 的概率是 59814，因此密文有 $1/10^5$ 的概率为 00000 97414。然而，如果明文换成 40186 37644，那么不存在可以产生密文 00000 97414 的密钥，所以这个密文发生的概率是 0。因此，如果 Eve 恰巧知道了这个密文，她就知道明文不是 40186 37644。 ■

6.3 完美保密与唯一解密性

令 $f(clear, key)$ 为一个密码系统的加密函数。我们已经知道，如果对于每个明文 i，$f_{plain=i}$是一个可逆函数，那么这个密码系统是完美保密的。

回想一下，一个密码系统满足唯一解密性意味着对于每个密钥，函数 f 的定义域中的每个元素恰巧是一个明文的加密。唯一解密性的另一种表达方式是，对每个密钥 k，$f_{key=k}$是一个可逆函数。

我们发现，判断一个密码系统是否满足唯一解密性与判断它是否满足完美保密性非常相似。唯一的区别是，一个是固定明文以密钥作为变量得到函数，另一个是固定密钥以明文作为变量得到函数[⊖]。从这个相似性的视角，我们说一个加密函数是唯一可解密钥的（uniquely de-keyable），如果对于每个 i，函数

⊖ 当然存在不可逆的完美保密的密码系统。设计一个！

$f_{plain=i}$是可逆的。

6.4 完美保密简史

6.4.1 弗纳姆机器

1917年，AT&T公司27岁的工程师吉尔伯特·弗纳姆发明了一个对电报信息加密/解密的机器。约瑟夫·莫博涅（陆军少校，通信研究工程部部长）考虑如何用它制作一个不会被攻破的系统。大概30年后，克劳德·香农提出了完美保密理论，并且用该理论证明了弗纳姆密码确实实现了完美保密。正如我
70 们将看到的，这种安全性是要付出相应代价的。

弗纳姆的发明的基础是模2加法。正如我们在例2a中看到的，通过模2加法进行加密可以实现完美保密性。模2运算只作用于1和0（被称作二进制数或比特），所以非常适合在数字电子的二进制世界实现。模2加法在密码学应用中有一个额外的优势。因为−1模2的代表元等于1，所以模2减1与模2加1是相同的。当然，减0与加0也是相同的。由此得出结论，解密函数（模2减密钥）与加密函数（模2加密钥）是相同的。

当模数是2时，只有两个可能的明文1和0。为了加密一个更长的明文，我们可以把它表示成0和1组成的字符串（称作二进制数或比特），并为每一个比特使用一个新的随机密钥，逐比特加密。所以密钥所需的比特数量就是要加密的明文的比特数量。

手工将自然语言下的明文转化为二进制并且再转化回来是一个烦琐的工作。当弗纳姆从事密码学工作的时候，一种将这些过程自动化的机器（电传打字机）就已经存在了。

电传打字机中使用的“摩斯”码是博多码，博多码以发明者J. M. E. Baudot的名字命名，每个符号都用五个比特组成的串表示。有32个这种串，所以有32个符号可以用这种方式表示。电传打字机将按键转化为通过一根导线传输的、表示1和0的电脉冲。这个机器也可以读穿孔纸带，在一条纸带上，用穿孔的有无代表1和0组成的长序列。

弗纳姆提出使用电传打字机将来自穿孔纸带的比特与按键产生的比特组合。每次按下一个键，就会产生与被按下符号对应的5个比特。同时，纸带向前移动5个位置，并从纸带读入5个比特。将按键产生的第一个比特与从纸带读入

的第一个比特相加（模 2），得到的结果比特通过线路传送。按键产生的其余 4 个比特中的每一个与从纸带读入的其余 4 个比特中的每一个以同样方式组合。通过使用纸带存储一个足够长的密钥比特序列，人们可以自动加密一个完整信息。

在另一端，拥有同一纸带的一个相似的机器可以实现解密。通过线路收到的比特与相应的存储在纸带上的比特进行模 2 加运算。

弗纳姆机器的一个重要优势在于它将加密过程与传输过程相结合。在这之前，我们必须首先加密信息，然后再单独传送密文。弗纳姆去掉了中间步骤，提升了效率并且降低了错误的可能性。 71

密钥纸带是成对产生的。为了产生五位比特的密钥，首先随机选取一个符号（从帽子里抓阄），按下相应的键，相应的 5 个比特就会被打孔在两个纸带上。然后选取另一个符号，以此类推。制作密钥纸带仍然是一个有点艰苦的过程。

在早期，密钥纸带形成一个循环，以便提供一个无限的密钥比特流。只要加密机器的纸带和解密机器的纸带相同，并且是在同一个地方开始的，它们就会永远保持同步。

然而，因为密钥最终会重复，所以密钥的循环不能提供完美保密性。莫博涅意识到任何有规律的重复都会导致不安全。更重要的是，他意识到如果纸带不循环，即如果每个随机密钥只被使用一次，这个系统将会满足完美保密性。即使窃听者有前半段明文的先验知识，并且窃听到了后半段的密文，这也对她没有任何帮助，因为后半段使用的密钥与前半段使用的密钥没有关系。

不管 AT&T 怎样努力推广弗纳姆机器，它仍然没有实现商业上的成功。企业更倾向于使用他们知道的方法。甚至美国政府拒绝使用不可攻破的密码系统，直到第二次世界大战来临。

6.4.2 一次性密码本

大约与弗纳姆和莫博涅在美国提出无法破解的密码系统的同一时间，在德国也发现了同样的思想。德国外交部的三个密码学家沃纳·昆茨、鲁道夫·谢弗和埃里希·朗洛茨偶然想到使用随机并且不重复的密钥。他们使用模 10 加法，而不是模 2 加法。此外，他们的系统并没有整合为一台机器，他们仍坚持雇职员手工执行加密和解密操作。与弗纳姆不同的是，政府使用了他们的系

统：大约 1922 年，德国外交部开始使用该系统。

密钥没有存储在纸带上，而是存储在密码本上。每个密码本包含 50 个编号的工作表，每张工作表为法律用纸尺寸大小，表上列有 48 个五位数。密码本是成对制作的，所以每个密码本仅有两份副本（copy）。一个副本被保留在
72 柏林，而另一个副本提供给大使馆。办事员一次使用密码本中的一张表，该表使用后即被销毁，所以一张表不会被再次使用。这个系统以一次性密码本而闻名。

6.5 完美保密密码系统的缺点

如果一次性密码本是完美保密的，政府为什么会使用别的系统呢？这是因为它带来完美保密性特征的同时也使系统变得不实用：它消耗了巨大数量的密钥。需要加密的信息流是巨量的，特别是在战争时期尤为如此，而制造密码本跟不上这个速度。

此外，假设许多地理位置分散的团体（假设是军团的通信职员）相互之间需要安全地通信。如果使用传统的密码系统，他们可能被提供一个密钥用于所有的通信。而若他们使用一次性密码本系统，有两种可能性，第一种是每一方与其他各方都分别有一个用来通信的不同的密码本。这个方法极大地增大了需要制造和记录的密钥数。第二种可能是各方都使用同一个密码本的副本。然而，在这种情况下，他们必须不断地彼此通信进行协调，来保证没有两方独立地使用密码本中的同一张密码表。例如，假设有四个团体 A、B、C 和 D，A 与 B 通信，C 与 D 通信。A 和 C 必须进行协调，以便 A 和 C 不会无意中使用了密码本中的同一张表。这种协调非常困难，特别是在战争时期（当其中一方不可达，你该怎么办?）。

1930 年，苏联在他们的军队、外交、商业、情报及反情报的通信中采用了一次性密码本系统。然而，在第二次世界大战初期，他们陷于密码本短缺。结果，他们被迫错误地使用系统：多个信息使用相同的密钥加密。这种情况可能由于以上的多方场景而发生，或许只是简单地因为制作密码本的速度限制。事实上，这个错用并不一定意味着违反了安全性。例如，典型地，一条用密钥加密的信息是相当低级别的外交信息，而另一条是与间谍相关的顶级机密的信息。大概一些密码职员推断这样使用可能更有可能逃过其他国家情报服务的注意。不管原因为何，这种（完美）安全性的暂时失效导致美国和英国都试图对

苏联和其代表处之间通信的数千条消息进行密码分析。这些信息中，只有1%产生威胁，并且在这1%中，只有几个单词被泄露。然而，这足以了解一些西 73
方感兴趣的主题，即苏联间谍对英国的情报部队——军情五处的渗透。

6.6 思考题

1. 当你在一个跑马场，一个骑师朋友悄悄给你一张纸，纸上有昨天五场预赛中参加今天比赛的四匹马的比赛结果。不幸的是，你忘记了之前商定的密钥是什么（但是你记得密钥是均匀随机选择的）。你知道信息是使用下面的方法加密的：

		明文（马匹）			
		0（幸运符）	1（闪电）	2（八号球）	2（天哪）
密钥	0	0	1	2	3
	1	1	2	3	0
	2	2	3	0	1
	3	3	0	1	2
	4	2	3	0	1
	5	2	3	0	1
	6	2	3	0	1
	7	2	3	0	1

考虑一下解密他的消息“33333”后，你能得到什么信息。

(a) 根据这条密文，获胜的马的组合可能是什么？

(b) 在今天的比赛中你对哪匹马下注？为什么？

2. 想象一下你是Eve，一天早上醒来，了解到Alice向Bob发送了一条消息。这条消息只有一个数字，这个数字是明文加上一个随机选择的密钥（模10）得到的。

(a) 据你目前所知，可能的明文是什么？

(b) 假设一个小时之后，你获得了密文。现在与你所知相一致的明文是什么？

(c) 现在假设第二天你醒来，了解到Bob向Alice发送了一条消息，这条消息包含两个数字，是通过将随机选择的一个密钥与每个数字加起来（模10）加密得来。根据你现在所知，可能的明文是什么？ 74

(d) 又一次，在思考了一个小时后，你观察到密文本身。现在与你所知相一致的明文是什么？

对于下面思考题3到思考题8描述的每个加密函数，回答下面的问题。

(i) 对于每个密钥，加密函数是唯一可解密的吗？如果是，使用法则定义解密函数。如果不是，给出一个密钥以及两个映射到同一个密文的明文。

(ii) 对于每个明文，加密函数是唯一可解密钥的吗？如果是，证明对于任意明文，密钥与密文之间的映射函数是可逆的（通过给出逆函数的法则）。如果不是，给出一个明文以及两个映射到同一个密文的密钥。

(iii) 如果（ii）中表明了加密方法不是唯一可解密钥的，那么它是完美保密的吗？如果是，在假设密钥是均匀随机选取的前提下，画出密文的概率分布。一定要标明坐标轴来说明概率值是什么。如果不是，给出两个不同的明文，这两个明文对应的密文的概率分布是不同的。

例子：

- 法则：$encrypt(clear,key)=clear+key \text{ rem } 10$
- 明文空间：整数 0 至 9
- 密钥空间：整数 0 至 9
- 密文空间：整数 0 至 9

(i) 是的，这个函数对于每个密钥都是唯一可解密的。解密函数的法则是 $decrypt(cyph,key)=cyph-key \text{ rem } 10$。

(ii) 是的，这个函数对于每个明文是唯一可解密钥的。解密函数的法则是 $key=cyph-clear \text{ rem } 10$。

(iii) 如（ii）中说明，这个函数是唯一可解密钥的。

例子：

- 法则：$encrypt(clear,key)=clear \cdot key \text{ rem } 10$
- 明文空间：整数 0 至 9
- 密钥空间：整数 0 至 9
- 密文空间：整数 0 至 9

(i) 这个函数对于每个密钥不都是唯一可解密的。例如，如果密钥是 5，明文 6 和 8 对应的密文都是 0。

(ii) 这个函数也不是唯一可解密钥的。如果明文是 2，使用密钥 3 和密钥 8 都可以得到密文 6。

75

(iii) 明文 3 和 4 有不同的密文概率分布。

例子：

- 法则：$encrypt(clear,key)=(clear \cdot key^2) \text{ rem } 7$

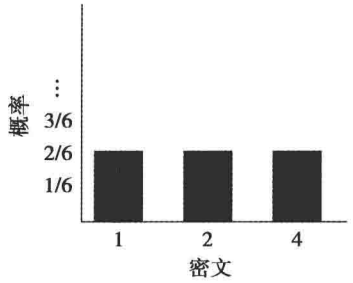
- 明文空间：1，2，4
- 密钥空间：1，2，3，4，5，6
- 密文空间：1，2，4

(i) 是的，这个函数对于每个密钥都是唯一可解密的。解密函数的法则是

$$decrypt(cyph, key) = \left(\frac{cyph}{key^2}\right) \text{ rem } 7$$

(ii) 不是，这个函数不是唯一可解密钥的。如果明文是 1，使用密钥 3 和 4 都可以得到密文 2。

(iii) 尽管这个函数不是唯一可解密钥的，但是它是完美保密的。概率分布是均匀的。每个可能的密文的概率都是 1/3。



3. 法则：$encrypt(clear, key) = clear \cdot key \text{ rem } 11$

   明文空间：整数 1 至 10

   密钥空间：整数 1 至 10

   密文空间：整数 1 至 10

4. 法则：$encrypt(clear, key) = clear^2 + key \text{ rem } 13$

   明文空间：整数 0 至 12

   密钥空间：整数 0 至 12

   密文空间：整数 0 至 12

5. 法则：$encrypt(clear, key) = clear + key^2 \text{ rem } 7$

   明文空间：整数 0 至 6

   密钥空间：整数 0 至 6

   密文空间：整数 0 至 6

6. 法则：$encrypt(clear, key) = 4 \cdot clear + key^2 \text{ rem } 14$

   明文空间：整数 0 至 13

   密钥空间：整数 0 至 13

密文空间：整数 0 至 13

7. 法则：$encrypt(clear, key) = (clear \cdot key^2) \text{ rem } 5$

明文空间：1，4

密钥空间：1，2，3，4

密文空间：1，4

8. 法则：$encrypt(clear, key) = clear + key$

明文空间=0，1，2，3

密钥空间=0，1，2，3

密文空间=0，1，2，3，4，5，6

下面的表表示加密每个信息分组的不同的加密函数。每个分组使用的密钥是均匀随机选择的。对于每个加密函数，请简要回答下面两个问题：

(a) 它有没有实现完美保密性？如果没有，为什么没有？

(b) 它有没有实现唯一解密性？如果没有，为什么没有？

9.

明文 \ 密钥	A	B	C	D
1	?	!	#	@
2	!	#	@	?
3	#	@	?	!
4	@	?	!	#

10.

明文 \ 密钥	A	B	C	D
1	?	!	#	@
2	!	?	@	?
3	#	@	!	?
4	@	#	?	!

11.

明文 \ 密钥	A	B	C	D
1	!	!	!	!
2	?	?	?	?
3	#	#	#	#
4	@	@	@	@

12.

明文 \ 密钥	A	B	C	D
1	!	?	#	@
2	?	#	@	!
3	#	@	!	?

77

13.

明文 \ 密钥	A	B	C	D
1	#	@	!	?
2	@	!	?	#
3	!	?	#	@
4	?	#	@	!
5	!	@	?	#

14. Abbot 和 Costello 为 NSO 效力，负责评价密码系统的安全性。他们收到一个加密函数的提案，如下表所示。

		密钥			
		A	B	C	D
明文	1	@	$	%	#
	2	#	%	$	@
	3	%	#	@	$
	4	$	@	#	%

Abbot 说：

“这个系统有如下性质：对于每个密钥，一个按照均匀分布（在所有可能的明文集合上）随机选取的明文被加密成一个按照均匀分布（在所有可能的密文集合上）随机选取的密文。因此，这个系统是完美保密的。”

Costello 回答说：

“Abbot，你错了。这个系统实现了完美保密性，但不是因为你给出的原因。事实上，我可以重新整理表格里的密文，使得加密方法具备你提出的性质，但是不具备任何安全性！”

Costello 是正确的。

(a) 说明如何重新整理表格里的密文以得到一个加密函数，该函数具备 Abbot 提出的性质，但却明显是不安全的。 78

		密钥			
		A	B	C	D
明文	1				
	2				
	3				
	4				

(b) 为什么一开始提出的加密方法实现了完美保密性？通过引用具体的表来解释一下。

15. 考虑下面的加密函数：

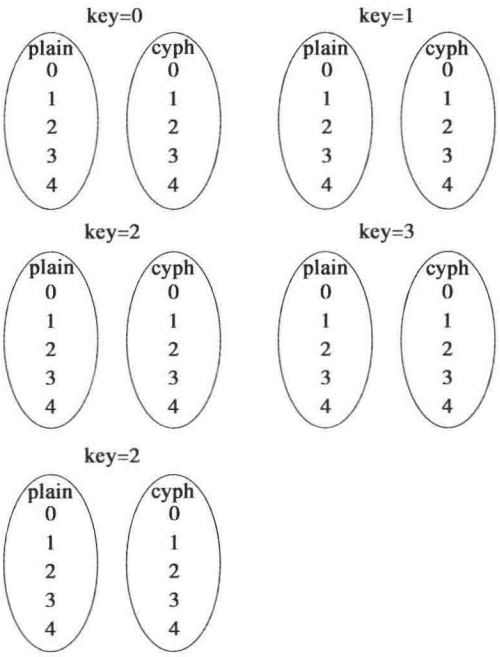
明文空间：0，1，2，3，4

密钥空间：0，1，2，3，4

密文空间：0，1，2，3，4

$$f(plain, key) = plain^3 + key^2 \text{ rem } 5$$

(a) 对于每个可能的密钥 k，相应加密函数的特殊化为 $Encrypt_{key=k}$，其定义域是明文集合，上域是密文集合。在下面简要画出这些特殊化函数。



79

(b) 这个函数是唯一可解密的吗?

(c) 这个函数是唯一可解密钥的吗?

(d) 这个加密函数是否实现了完美保密性?如果实现了,解释一下为什么;如果没有实现,提出一个具有相同密钥空间、明文空间、密文空间并且实现完美保密的函数。

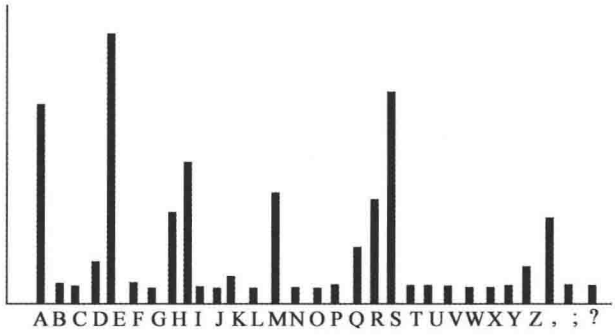
16. 每天深夜两点,《波士顿环球报》将用于第二天早报的文本通过电话线传送给印刷机构。文本包含 50 000 个符号,每个符号用一个 0 到 29 之间的数字表示。因此,这个文本是由 50 000 个数字组成的序列表示的。为了安全起见,《波士顿环球报》使用一次性密码本加密这个序列。每个密文数字是相应的明文和密钥的模 30 相加的和。

《普罗维登斯日报》听到传闻说《波士顿环球报》将要发表一篇讲述爱国者队为什么不会搬到罗德岛的真实原因的文章。《普罗维登斯日报》想要得到这篇文章并且赶在《波士顿环球报》之前发布。他们雇用了以窃听技巧闻名的 Eve,让她在《波士顿环球报》的密文发送到印刷机构的途中截获该信息。Eve 截获了密文并且通过调制解调器发送给了《普罗维登斯日报》的秘密部门。

Eve 接受这份工作但是并不想爬进下水道,也不想爬上电线杆(不想

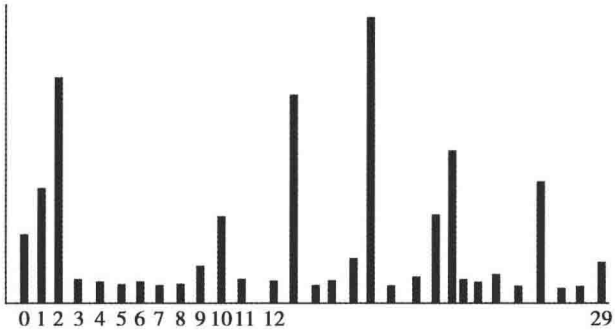
辛苦地工作)。因此，她设计了一个方案来欺骗《普罗维登斯日报》。她伪造了一个密文发送给《普罗维登斯日报》，那样，她就可以温暖舒服地待在家里了。

她从以前几期《波士顿环球报》中了解到符号的分布如下图所示。



80

她想出下列的数字分布：



她生成一个 50 000 个数字的序列，每个数字是按上述分布挑选的。然后她把这个序列发送给《普罗维登斯日报》。

(a) 为什么《普罗维登斯日报》意识到她试图欺骗他们？准确、具体地解释一下。

(b) Eve 怎么能够成功欺骗《普罗维登斯日报》？准确、具体地解释一下。

81

第7章
A Cryptography Primer: Secrets and Promises

数　论

7.1 整除

如果 c/b 是一个整数，我们就说整数 b 均分（evenly divide）整数 c。实际上，在数学上没有人说 b “均分” c——人们只是说 b “整除”（divide）c。另一种说法是 b 是 c 的一个因数（因子），c 的因数是能够整除 c 的数。最后，也可以说 c 可以被 b 整除。

例如：

- 3 整除 12
- 3 是 9 的一个因数
- 40 不是 20 的因数
- 40 可以被 20 整除
- 4 整除 4
- 5 是 −10 的一个因数
- 12 整除 60
- 50 的正因数是 1，2，5，10，25 和 50

7.2 互素

称两个数 r 和 s 是*互素*的，如果没有大于 1 的正数既是 r 的因数也是 s 的因数。在这种情况下我们还说 r 与 s 互素。例如，18 和 8 不互素，因为 2 既是 18 的因数也是 8 的因数（称为公因数）。另一方面，9 和 8 是互素的，因为它们两个仅有的共同的因数（公因数）是 1 和 −1。当确定互素的关系时，我们从来不考虑 1 和 −1 是它们的公因数。

- 20 和 40 不是互素的 —— 例如，20 是一个公因数。
- 27 和 80 是互素的。27 的所有大于 1 的因数是 3，9 和 27，都是 3 的幂，

并且 3 不能整除 80。

- 17 与 27 是互素的。
- 17 和 0 不是互素的，因为 17 是这两个数的公因数。
- 1 和 6 是互素的，因为 1 仅有的正因数是 1。

7.3 素数

如果一个大于 1 的正整数 n 仅有 1 和它本身两个正因数，则称之为*素数*。因此前几个素数是 2，3，5，7，11，13，17，19，23，29 和 31。仅有的偶素数是 2，因为其他所有的偶数都有 2 作为一个因数。我们不将 1 计入素数。

如果 n 是一个素数。什么数与 n 互素呢？因为 n 仅有的大于 1 的因数是 n 本身，所有不能被 n 整除的数都与 n 是互素的。能够被 n 整除的非负整数是 0，n，$2n$，$3n$，等等。

7.4 素因子分解

60 的分解是确定一些素数使它们的乘积是 60 的过程。很明显 6 是一个因数，$6 \cdot 10 = 60$。但 6 和 10 不是素数，所以我们必须进一步分解这些数。6 是素数 2 和 3 的乘积，10 是素数 2 和 5 的乘积。我们有

$$6 = 2 \cdot 3$$

$$10 = 2 \cdot 5$$

将这些等式相乘在一起，得到

$$6 \cdot 10 = 2 \cdot 3 \cdot 2 \cdot 5$$

所以 60 的素因子分解是 2，2，3 和 5。注意列表中 2 出现了两次，因为 2 要乘两次。

现在我们用不同的方式分解 60。我们观察到 60 是偶数，因此 2 是一个因数。用 60 除以 2，得到 30。现在 30 仍然是一个偶数，因此 2 是一个因数。除以 2，得到 15。最后，15 是 3 和 5 的乘积。所以素因子分解是 2，2，3 和 5。

数论中的基本结论是一个正数的素因子分解不依赖于这些因数是怎么找到的，一个正数有且仅有一种素因子分解。

7.5 欧拉函数 $\phi(x)$

数学家莱昂哈德·欧拉定义了函数 $\phi(x)$=小于 x 的数中与 x 互素的整数的个数。在本书中，我们使用了两个关于 $\phi(x)$ 的公式：

如果 m 是素数，那么 $\phi(m)=m-1$。

因为 1 至 $m-1$ 的每个整数都与 m 互素，所以这个公式成立。

若 $m=p\cdot q$，其中 p，q 是两个不同的素数，那么 $\phi(m)=(p-1)\cdot(q-1)$。

这个公式成立，因为 0 到 $m-1$ 中的数中，除了下面的数都与 m 互素。

$$0\cdot p,1\cdot p,2\cdot p,\cdots,(q-1)\cdot p$$

和

$$0\cdot q,1\cdot q,2\cdot q,\cdots,(p-1)\cdot q$$

第一列中有 q 个数，第二列中有 p 个数，总共有 $q+p$ 个数。因为 0 出现在两个列中，所以真实的总数是 $q+p-1$。因此，0 至 $m-1$ 之间与 m 互素的数的个数等于 0 到 $m-1$ 之间数的总数（也就是 m）减去与 m 不互素的数的个数（也就是 $q+p-1$）。我们得到

$$\phi(pq)=pq-(q+p-1)=(p-1)\cdot(q-1)$$

当 p 和 q 是比较小的数时，$\phi(pq)$ 明显小于 pq。例如，$\phi(6)=(2-1)\cdot(3-1)=2$。再如：$\phi(15)=8$。

然而，当 p 和 q 非常大时，pq 是巨大的，所以 pq 和 $pq-(q+p-1)$ 之间的差距就没有那么显著了。例如，假设 $p=9871$，$q=9533$。那么 $pq=94\ 100\ 243$，$\phi(pq)=94\ 080\ 840$。如果你在 0 至 $pq-1$ 之间随机选一个数，
84 那它很可能与 pq 互素。

7.6 乘幂

在讨论使用欧拉函数之前，先简单回顾一下乘幂。我们用过 10^{20} 这种表达式，读者很可能记得这个表达式代表

$$\underbrace{10\cdot10\cdot10\cdot10\cdot10\cdot10\cdot10\cdot10\cdot10\cdot10\cdot10\cdot10\cdot10\cdot10\cdot10\cdot10\cdot10\cdot10\cdot10}_{20\text{次}}$$

底数（在这个例子中是 10）表示什么数多次乘以它本身，指数（在这个例子中是 20）表示底数乘以它本身的次数。

7.6.1　幂指数相加法则

我们可以分开这个乘法，因此：

$$\underbrace{10 \cdot 10 \cdot 10 \cdot 10 \cdot 10}_{5\text{次}} \cdot \underbrace{10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10}_{15\text{次}}$$

这说明

$$10^5 \cdot 10^{15} = 10^{20}$$

这是乘幂的一个一般法则的例子：

幂指数相加法则　$b^{c+d} = b^c b^d$

简言之，幂指数相加相当于幂表达式相乘。

7.6.2　幂指数相乘法则

我们也可以把这个乘法分成五组，每组四个数相乘：

$$\underbrace{10 \cdot 10 \cdot 10 \cdot 10}_{4\text{次}} \cdot \underbrace{10 \cdot 10 \cdot 10 \cdot 10}_{4\text{次}} \cdot \underbrace{10 \cdot 10 \cdot 10 \cdot 10}_{4\text{次}} \cdot \underbrace{10 \cdot 10 \cdot 10 \cdot 10}_{4\text{次}} \cdot \underbrace{10 \cdot 10 \cdot 10 \cdot 10}_{4\text{次}}$$

可以重写为

$$10^4 \cdot 10^4 \cdot 10^4 \cdot 10^4 \cdot 10^4$$

我们注意到 10^4 乘以它自身五次。这相当于使 10^4 乘了五次，也就是 $(10^4)^5$。这说明

$$10^{20} = (10^4)^5$$

这是乘幂的另一个一般法则的例子：

幂指数相乘法则　$b^{c \cdot d} = (b^c)^d$

85

简言之，幂指数相乘相当于用底数与第一个指数做幂运算，然后将结果与第二个指数做幂运算。

7.7　欧拉定理

现在我们给出一个定理，它对第 14 章描述的密码方案非常重要。

欧拉定理：对任意模数 m 以及与 m 互素的任一整数 b，

$$b^{\phi(m)} \equiv 1 \pmod{m}$$

我们可以使用这个定理来简化乘幂表达式。例如，令 $m=4001$，这是一个素数。那么 $\phi(m)=4000$。令 b 是一个小于 4001 的正数。那么

$$b^{8003} = b^{2\cdot 4000+3} = (b^{4000})^2 b^3$$

这里用到了 7.6 节描述的乘幂的两个法则。

因为 $b^{4000} \equiv 1 \pmod{4001}$，我们可以用 1 代替 b^{4000} 来简化模 4001 的表达式。

$$(b^{4000})^2 b^3 \equiv (1)^2 b^3 \equiv b^3 \pmod{4001}$$

类似地，可以得到

$$b^3 \equiv b^{12003} \equiv b^{16003} \equiv b^{20003} \pmod{4001}$$

等等。确切的指数不重要，重要的是指数 mod $\phi(m)$ 的代表元。

7.8 思考题

1. 对于下面的每一数对，说出这两个数是否互素。
   (a) 18，4
   (b) 7，27
   (c) 24，33
   (d) 22，51
   (e) 0，17
2. 对于以下数字 n，列出所有小于 n 且和 n 互素的非负整数，并使用该列表计算 $\phi(n)$。
   (a) 18
   (b) 23 86
   (c) 77
   (d) 16
   (e) 15
3. 使用乘幂法则对以下每个公式进行简化。
   (a) $\dfrac{x^{15}}{x^7 / (xy)^8}$
   (b) $(x^{20})^{(y^5 y^{10} - y^{15})}$

(c) $y^{15}-\dfrac{y^4 y^6}{y^3 y^8}$

在接下来的思考题中，你可以使用欧拉定理和你知道的模算术的知识来简化模幂表达式 $b^t \text{ rem } m$，使其幂指数是一个小的非负整数。应该假设底数 b 与模数 m 互素，以便可以使用欧拉定理。

例子：

问题：模数 $m=4001$（一个素数）。化简 $b^{12006} \text{ rem } m$。

答案：$\phi(m)=4000$。简化如下：

$$b^{12006}=b^{3\cdot 4000+6}=(b^{4000})^3 b^6$$

由欧拉定理有 $b^{4000}\equiv 1 \pmod m$，我们可以化简 $b^{12006} \text{ rem } m$ 到 $(1)^3 b^6 \text{ rem } m$，也就是 $b^6 \text{ rem } m$.

4. 在本问题中使用模数 $m=17$。

(a) 化简 $b^{19} \text{ rem } 17$

(b) 化简 $b^{33} \text{ rem } 17$

(c) 化简 $b^{52} \text{ rem } 17$

(d) 化简 $b^{213} \text{ rem } 17$

5. 本问题中使用模数 $m=61$。

(a) 化简 $b^{61} \text{ rem } 61$

(b) 化简 $b^{185} \text{ rem } 61$

(c) 化简 $b^{2410} \text{ rem } 61$

6. 在本问题中使用模数 $m=143$。

(a) 化简 $b^{225} \text{ rem } 143$

(b) 化简 $b^{481} \text{ rem } 143$

(c) 化简 $b^{12037} \text{ rem } 143$

7. 下面的等式中哪些是正确的，哪些是错误的？

(a) $b^{21}\equiv b^4 \pmod{17}$?

(b) $b^{28}\equiv b^6 \pmod{23}$?

(c) $b^{59}\equiv b^{125} \pmod{67}$?

(d) $b^{540}\equiv b^{77} \pmod{463}$?

(e) $b^{723}\equiv b^5 \pmod{719}$?

8. 求出 s 或解释一下为什么解不存在。

(a) $(b^5)^s \equiv b \pmod{35}$

(b) $(b^6)^s \equiv b \pmod{23}$

(c) $(b^{101})^s \equiv b \pmod{7}$

88 (d) $(b^7)^s \equiv b \pmod{33}$

第8章

A Cryptography Primer: Secrets and Promises

欧几里得算法

在第4章的4.3.3节中，我们介绍了*模乘法逆元*的概念：对于模数 m，如果两个整数 a，b 满足 $a \cdot b \equiv 1 \pmod{m}$，则称 a 和 b 模 m 乘法互逆。在4.3.4节，我们演示了一种发现模 m 乘法逆元的方法：对于模 m 的所有代表元，写下这些元素之间的模 m 乘法表，然后在其中搜索运算结果为1的内容。在4.3.5节中，我们发现，对于模数6，有些整数并没有乘法逆元，甚至是一些模6不同余于0的数也没有乘法逆元。我们在本章中提到的方法将解释这种现象。

在本章中，我们描述了一个计算模乘法逆元的好的算法。该算法是对生活在公元前300年的古典数学家欧几里得（Euclid）做出的一个贡献的扩展。欧几里得广为人知的主要贡献集中在几何学的系统化上，但他著名的著作《几何原本》中也阐述了数论方面的知识。

8.1 测量谜题

假设给你两个容器，并且告诉你每个容器可以容纳多少杯水。也给你一个空水池，可以容纳无限量的水。使用这两个容器在水池中注入一些水然后取出，问：你可以在水池中留下的水的最小杯数（正整数）是多少？此谜题允许你使用这两个容器从水龙头或者水池中取水，可以把容器中的水倒在下水道或者水池中。

比如，对于一个7杯容量的容器和一个5杯容量的容器，你可以在水池中只留下一杯水。首先，使用7杯容量的容器从水龙头取水并将水倒入水池中，连续做三次，现在水池中多了21杯水。然后，使用5杯容量的容器从水池中取水并将水倒入下水道，连续做四次。这样水池中少了20杯水。最终在水池中留下了1杯水。如图8.1所示。

对于喜欢几何学的人，我们可以用几何方面的术语来解释这个谜题。给你两根棍子，并告诉你每根棍子的长度是多少英寸[⊖]。你想测量一个尽可能小

⊖ 1英寸=0.0254米。

（正的）的长度。重新阐述上边的例子，给你一个 7 英寸的棍子和一个 5 英寸的棍子，你可以测量出 1 英寸的长度。正像图 8.2 描述的一样，你可以使用 7 英寸的棍子去发现距离起始点向右 21 英寸的点 p，然后用 5 英寸的棍子去发现距离点 p 向左 20 英寸的位置。

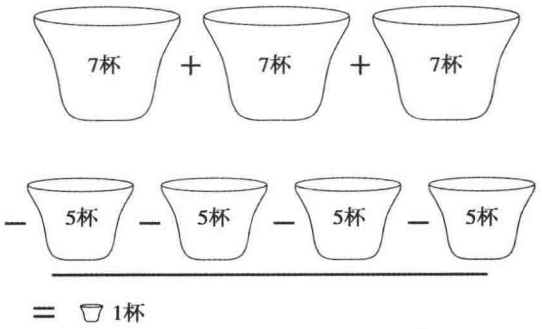


图 8.1 使用容量为 7 杯和 5 杯的容器的谜题解决方案

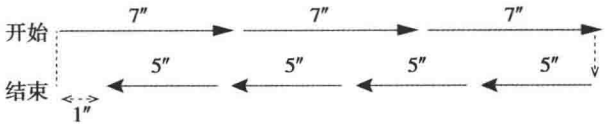


图 8.2 棍子版本而非杯子版本的相同谜题

注意 1 杯水（或 1 英寸）是我们所能希望的最佳结果。假设每个容器可以装下整数杯水（我们说容器的容量是整数，意思是每一杯水都是完整的整体，而且可以使用每个容器整数次）。因此整个过程中，水池中的水量保持整数杯水。最小的正整数当然是 1。

这个谜题的解决方法不仅仅是最终数量“1”，重要的是解决该问题的方法（加 7 杯 3 次，然后减 5 杯 4 次）。这个方法可以使用一个简洁的数学公式来表示：

$$1 = 3 \cdot 7 - 4 \cdot 5$$

该等式表示可以进行多少次的加 7 操作（3 次）和多少次的加 5 操作（−4 次）。

90

8.1.1 一个更复杂的例子

假设给定一个容量为 524 杯的容器和一个容量为 876 杯的容器。你最少可以在水池中留下多少杯水呢？我们之前说过，整个过程中水池中的水保持整数杯。

我们使用同样的方法讨论，对于这样容量的两个容器，水池中能留下水的杯数是4的倍数。

开始水池中有0杯水，0是4的倍数。每次用小容器向水池中倒水，会在水池中增加524杯水。注意，524是4的倍数，所以把一个4的倍数加到了4的倍数。同样的，每次用小容器从水池中取水的时候，便将水池中的水减掉了524杯，因此也就是从4的倍数中减掉了4的倍数。大容器的容量为876杯水，也是4的倍数，所以对于小容器的讨论同样适用于大容器注水和取水过程中。所以，不管你怎么做，只要遵守规则，在水池中的水将永远是4的倍数。

最小的4的正整数倍是4本身，所以我们能希望的最好的结果是留在水池中的水的杯数是4。实际上，我们可以精确地实现下述结果：使用小容器对水池进行107次注水，然后使用大容器从水池中取水64次，每次将大容器取出的水倒入下水道。（你可能说这是浪费水，但要实现精确必须要有代价。）在数学上，我们的技巧可以写作：

$$4 = 107 \cdot 524 - 64 \cdot 876$$

8.2 通过解决测量谜题求模乘法逆元

测量谜题和模乘法逆元有什么关系呢？

例1 假设模数是7，想计算5在模7下的乘法逆元。为了解决这个问题，拿一个容量为7杯水的容器和容量为5杯水的容器。从之前的结果我们知道，可以利用这样的容器在水池中留下1杯水。这种方法可以使用数学公式表达为： 91

$$1 = 3 \cdot 7 - 4 \cdot 5$$

两边同时减掉$-4 \cdot 5$，得到

$$1 - (-4 \cdot 5) = 3 \cdot 7$$

这个式子表示1和$-4 \cdot 5$的差是7的倍数，也就是说，

$$-4 \cdot 5 \equiv 1 \pmod 7$$

因此，-4是5关于模7的乘法逆元，成功了！ ■

例2 模数是782394，需要找到387451的模乘法逆元。考虑使用容量为782394和容量为387451的容器的测量谜题。结果表明最终可以在水池中保留一杯水，对应的公式是：

$$1 = -58490 - 782394 + 118111 \cdot 387451$$

这个等式表明，118111 • 387451 和 1 是同余的（因为它和 1 仅仅相差模数的倍数），因此 118111 是 387451 的模乘法逆元，又一次成功了！ ■

到目前为止我们运气还不错。接下来的例子就不像我们所预料的那样了。

例 3 模数是 876，要找到 524 的模乘法逆元，也就是找到整数 q，使得 $1-q \cdot 524$ 是 876 的倍数。“一个 876 的倍数”意味着某个整数乘以 876，让我们用 p 来表示这个整数。因此，为了成功，必须存在整数 p，q 满足 $1-q \cdot 524 = p \cdot 876$。是否存在这样的整数呢？

对等式两边都加上 $q \cdot 524$，得到

$$1 = p \cdot 876 + q \cdot 524 \tag{8.1}$$

存在满足该等式的整数 p 和 q 意味着有一种方法，使得使用容量为 876 和 524
92 的容器的测量谜题恰好只在水池中留下 1 杯水。我们在 8.1.1 节中已经说明这是不可能的。因此 524 没有关于模数 876 的乘法逆元。 ■

8.1.1 的讨论归纳如下。假设等式（8.1）对于某些整数 p 和 q 成立，因为 4 是 876 和 524 的因数，因而也是 $p \cdot 876 + q \cdot 524$ 的因数。既然等式（8.1）成立，那么 4 将是 1 的因数，所以说该等式不可能成立。

8.3 欧几里得算法

在本节中，我们将概述欧几里得算法及其用途。

8.3.1 欧几里得算法计算什么

对于任何输入的整数 a 和 b，欧几里得算法计算三个整数 s，t 和 d，使其满足三个特性：

$$d > 0 \tag{8.2}$$

$$d = s \cdot a + t \cdot b \tag{8.3}$$

$$d \text{ 是 } a \text{ 和 } b \text{ 的因数} \tag{8.4}$$

这些特性看起来非常普通，但是组合在一起却非常有力。它们可以使我们得到两个结论。我们现在将陈述这些结论以及相应的数学证明。尽力去跟上这些证明，但是看不懂时也不要担心。

结论 1 *d* 是整除 *a* 和 *b* 的最大整数。

证明 任何其他 *a* 和 *b* 的因数 *e* 同样也是 $s \cdot a$ 和 $t \cdot b$ 的因数，进而也是 $s \cdot a + t \cdot b$ 也就是 *d* 的因数。因为 *e* 是 *d* 的因数，且 *d* 是正数，我们可知 *e* 小于 *d*。

结论 2 *d* 是使用容量为 *a* 和 *b* 的容器的测量谜题中所能在水池中留下的最小正整数杯数。

证明 考虑任何使用容量为 *a* 和 *b* 的容器的倒水方案。假设使用了容量为 *a* 的容器进行了 *p* 次倒水，使用容量为 *b* 的容器进行了 *q* 次倒水（这里 *p* 和 *q* 可能是正整数、负整数或零）。那么该方案水池中留下的总水量为 $p \cdot a + q \cdot b$。 93

因为 *d* 是 *a* 的因数，所以也是 $p \cdot a$ 的因数。同样，因为 *d* 是 *b* 的因数，所以也是 $q \cdot b$ 的因数。既然 *d* 是 $p \cdot a$ 和 $q \cdot b$ 的因数，因而它也是它们之和 $p \cdot a + q \cdot b$ 的因数。这表示 *d* 是上面倒水方案在水池中最终留下水量的一个因数。因此 *d* 小于等于任何倒水方案在水池中留下的水的杯数。

因为结论 1，*d* 被称为 *a* 和 *b* 的最大公因数。欧几里得算法是用来寻找最大公因数的算法。然而，我们对欧几里得算法得到的 *s* 和 *t* 也感兴趣。

- 如果最大公因数是 1，那么等式（8.3）表示 *t* 是 *b* 关于模数 *a* 的乘法逆元。
- 如果最大公因数比 1 大，则结论 2 意味着没有整数 *p* 和 *q* 满足 $1 = p \cdot a + q \cdot b$，因此 *b* 没有关于模数 *a* 的乘法逆元。

在第 7 章中，我们介绍了一个关于整数 *a* 和 *b* 的最大公因数是 1 的术语：我们说 *a* 和 *b* 是互素的。因此，欧几里得算法告诉我们当 *a* 和 *b* 互素的时候，*b* 关于模数 *a* 的乘法逆元存在。

8.3.2 前向计算

我们用一个例子来说明欧几里得算法。假设有容量为 12 杯水和容量为 5 杯水的容器。我们可以让大容器像容量为 2 杯水的容器一样使用，也就是说，可以使用它向水池中注入或盛出 2 杯水。比如，想要倒入 2 杯水。

- 使用大容器从水龙头中取水。
- 使用小容器从大容器中取水，然后把小容器中的水倒入下水道。
- 再做一遍前一步。
- 现在大容器中有 2 杯水的剩余，然后把大容器中的水倒入水池中。

一个类似的过程可以被用在从水池中减少两杯水。因此，为了解决这个谜
94 题，我们可以想象有一个虚拟的容量为 2 杯水的容器和容量为 5 杯水的容器。让我们来解决这个新的谜题。

现在，我们来演示容量为 5 杯水的容器可以被当作容量为 1 杯水的容器使用（别忘了那个虚拟的容量为 2 杯水的容器）。

- 使用容量为 5 杯水的容器从水龙头中取水。
- 使用容量为 2 杯水的容器从容量为 5 杯水的容器中取水，然后把取到的水倒入下水道。
- 再做一遍。
- 现在容量为 5 杯水的容器中有 1 杯水。

当然，现在我们知道了如何把容量为 5 杯水的容器当作容量为 1 杯水的容器使用。此时解决测量谜题（在水池中留下最少的水）只需直接利用容量为 1 杯水的容器将一杯水倒入水池中即可。

让我们再尝试另一个例子。

- 假设我们开始从 44 杯水的容器和 13 杯水的容器开始。可以想象大容器是一个容量为 5 杯水的容器（我们将大容器装满水，然后使用小容器从大容器中取三次水，大容器中将剩余 5 杯水）。
- 现在我们需要解决一个容量为 13 杯水的容器和容量为 5 杯水的容器的谜题。可以把容量为 13 杯水的容器当作一个容量为 3 杯水的容器进行使用（将容量为 13 杯水的容器装满水，然后使用容量为 5 杯水的容器从其中取水两次）。
- 现在我们需要解决一个容量为 5 杯水的容器和容量为 3 杯水的容器的谜题。可以把容量为 5 杯水的容器当作容量为 2 杯水的容器（将容量为 5 杯水的容器装满水，然后从其中移出 3 杯水）。
- 我们需要解决容量为 2 杯水的容器和容量为 1 杯水的容器的谜题。这是显而易见的，因为这里我们有了一个 1 杯水的容器。

产生假想容器的规则是这样的：如果你有一个容量为 x 杯水的容器和容量为 y 杯水的容器且 $x>y$，可以想象你有一个容量为 y 杯水的容器和容量为 x rem y 杯水的容器（x rem y 是 x 除以 y 之后的余数）。接下来，对这两个新的容器使用同样的规则，将容量为 y 杯水的容器使用想象的新容器替换，依次这样处理，直到某个新的想象中的容器的容量为 0。此时与之相对应的容量大的容器的容量就是我们可以在水池中留下的水的最小杯数。

我们可以再看一个例子，但是这次用数字描述，而非杯子数。 95

1. 从 77 和 23 开始。77 除以 23 余数为 8。新得到的两个数为 23 和 8。

2. 23 除以 8 余数为 7。新得到的两个数为 8 和 7。

3. 8 除以 7，余数为 1。新得到的两个数是 7 和 1。

4. 7 除以 1，余数为 0。新得到的两个数是 1 和 0。

我们在此处停止，推出在水池中可以留下的最少水量是 1 杯。

8.4 欧几里得算法的后向部分

前一节描述的方法仅仅告诉我们如何去解决原始的测量谜题，并没有给出精确的公式来告诉我们需要利用最初容器的容量进行多少次加减操作。为了得到这个公式，我们需要对计算过程进行反推，这个过程需要执行一些代数运算。

作为第一个例子，我们重新回顾一下上一节最后一个例子。首先回顾之前的前向计算过程。每当提出一个新的假想的容器时，我们写下一个等式表示如何使用两个已经存在的容器来实现这个想象出来的容器。

前向计算

$$77-3\cdot 23=8，\text{所以接下来使用 23 和 8} \tag{8.5}$$

$$23-2\cdot 8=7，\text{所以接下来使用 8 和 7} \tag{8.6}$$

$$8-1\cdot 7=1，\text{所以接下来使用 7 和 1} \tag{8.7}$$

7 除以 1 的余数是 0，因此停止运算。

前向计算表明，我们可以最终在水池中留下 1 杯水。让我们推导一个公式来表示如何实现。

使用（虚拟的）容量为 7 杯水的容器和（虚拟的）容量为 1 杯水的容器，容易得到：

$$1 = 0 \cdot 7 + 1 \cdot 1$$

该等式表示我们向水池中添加 1 次 1 杯水和 0 次 7 杯水。为了得到使用容量为 8 杯水的容器和容量为 7 杯水的容器如何留下 1 杯水的等式，我们使用等式
96 (8.7) 来代替上一等式中的最后一个 1，其中 1 代表最小想象的容器的容量。

$$1 = 0 \cdot 7 + 1 \cdot (8 - 1 \cdot 7)$$

使用分配律得到：

$$1 = 0 \cdot 7 + 1 \cdot 8 - 1 \cdot 1 \cdot 7$$

合并乘以 7 的项得到：

$$1 = 1 \cdot 8 + (0 - 1 \cdot 1) \cdot 7$$

最后，化简括号中的表达式，最终得到：

$$1 = 1 \cdot 8 - 1 \cdot 7$$

(许多获得等式的工作我们早已知道，但是它阐明了一系列步骤中所需的计算)。我们现在有了一个等式，表明怎样使用 8 杯水的容器和 7 杯水的容器来留下 1 杯水。容量为 7 杯水的容器是我们想象中的，因此继续使用等式 (8.6) 对 7 进行代换，得到：

$$1 = 1 \cdot 8 - 1 \cdot (23 - 2 \cdot 8)$$

使用分配律得到：

$$1 = 1 \cdot 8 - 1 \cdot 23 - 1 \cdot (-2) \cdot 8$$

合并乘以 8 的项得到：

$$1 = -1 \cdot 23 + (1 + 2) \cdot 8$$

化简括号中的表达式得到：

$$1 = -1 \cdot 23 + 3 \cdot 8$$

我们有了一个等式，表明怎样使用 23 杯水的容器和 8 杯水的容器来留下 1 杯水。容量为 8 杯水的容器是我们想象中的，因此继续使用等式 (8.5) 对 8 进行代换，得到：

$$1 = -1 \cdot 23 + 3 \cdot (77 - 3 \cdot 23)$$

合并乘以 23 的项得到：

$$1 = 3 \cdot 77 + (-1 + 3 \cdot (-3)) \cdot 23$$

化简得到：

$$1 = 3 \cdot 77 - 10 \cdot 23$$

97 就是我们最终想要的等式。

让我们再看一个例子，这次减少解释。我们可以从容量为 876 和 524 开始。

前向计算

$$876-1\cdot 524=352 \tag{8.8}$$

$$524-1\cdot 352=172 \tag{8.9}$$

$$352-2\cdot 172=8 \tag{8.10}$$

$$172-21\cdot 8=4 \tag{8.11}$$

$$8-2\cdot 4=0$$

此时计算终止，因为余数已经是 0 了。

在 0 之前计算得到的最后一个数 4，是 876 和 524 的最大公因数。回顾一下，因为 4 比 1 大，我们可知 524 没有关于模 876 的乘法逆元。

后向计算

$4=0\cdot 8+1\cdot 4$　　使用等式(8.11)对 4 进行代换

$=0\cdot 8+1\cdot(172-21\cdot 8)$

$=1\cdot 172+(0-1\cdot 21)\cdot 8$

$4=1\cdot 172-21\cdot 8$　　使用等式(8.10)对 8 进行代换

$=1\cdot 172-21\cdot(352-2\cdot 172)$

$=-21\cdot 352+(1-21\cdot(-2))\cdot 172$

$4=-21\cdot 532+43\cdot 172$　　使用等式(8.9)对 172 进行代换

$=-21\cdot 352+43\cdot(524-1\cdot 352)$

$=43\cdot 524+(-21+43\cdot(-1))\cdot 352$

$4=43\cdot 524-64\cdot 352$　　使用等式(8.8)对 352 进行代换

$=43\cdot 524-64\cdot(876-1\cdot 524)$

$=-64\cdot +(43-64\cdot(-1))\cdot 524$

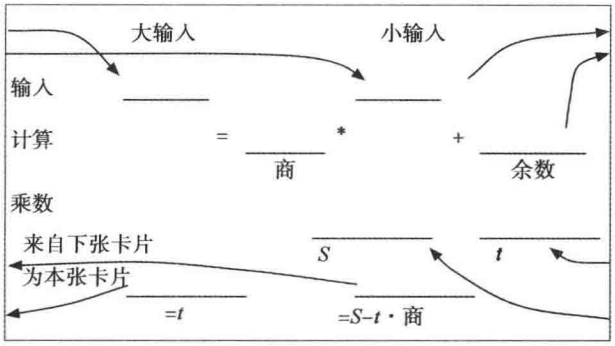
$4=-64\cdot 876+107\cdot 524$

8.5 欧几里得卡片

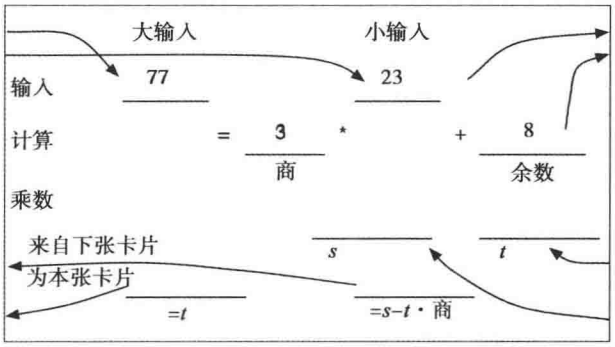
尽管欧几里得算法的步骤非常简单，但是非常容易让人疑惑。为了帮助你实现这些步骤，我们引入一些指导你行动的卡片。你可以在前向计算的每一步

98 中使用一张卡片。

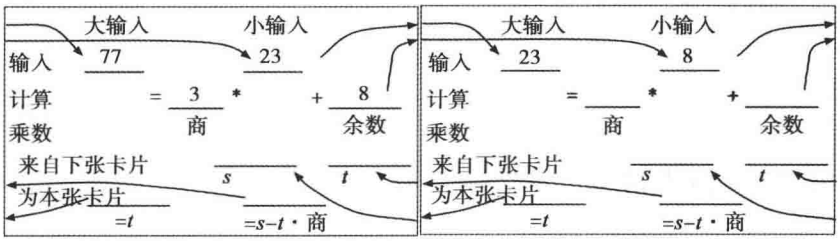
卡片看上去像下面的样子：



在卡片的顶部，有两个标注着大输入和小输入的空格。可以将容器的容量写在这些空格中（比如 77 和 23）。在下一行中，有两个空格可以填写商（由大输入被小输入除得到）和余数。这给出前向计算的一个等式，也就是 77＝3 · 23＋8，卡片变成如下样子。



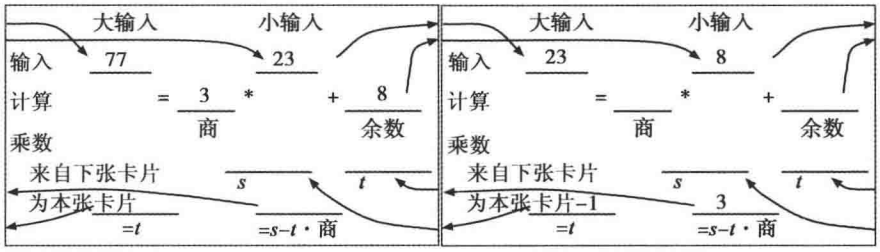
注意到箭头方向从卡片右边的小输入和余数开始。这些箭头表示这两个数应该被拷贝到另一张卡片上；新的卡片的左边应该连接到旧卡片的右边，这样就将箭头连接起来了。



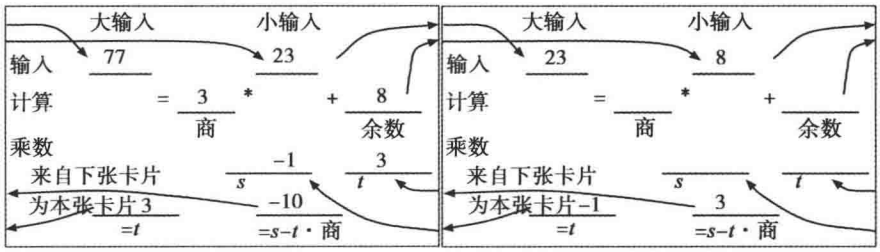
99

现在你已经准备好去求 23 除以 8 之后的商和余数，从而得到前向计算中的第二个等式。

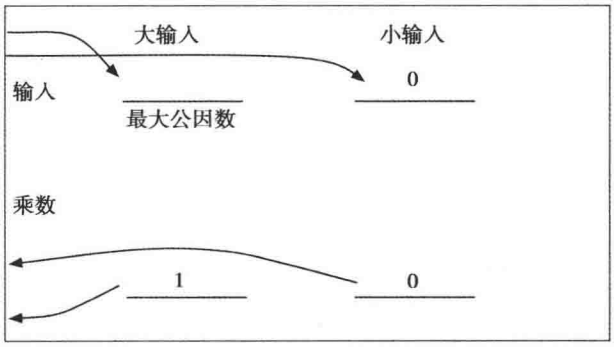
计算最终将得到 23 和 8 的乘数，即需要乘到 23 和 8 上以得到 1 的数字。它们最终可以写在相应卡片的最后一行。



在这个例子中，对于 23 和 8 的乘数分别是 −1 和 3。应该将这两个值拷贝到第一个卡片中标记为 s 和 t 的空格中。现在可以从 s 和 t 得到 77 和 23 的乘数。公式由卡片中给出：77 的乘数就是 t（这里是 3），23 的乘数是 $s-t\cdot$ 商，也就是 $(-1)-3\cdot 3$，因为在第一个卡片中商是 3。因此可以将 3 和 −10 写入第一个卡片中的最后一行。



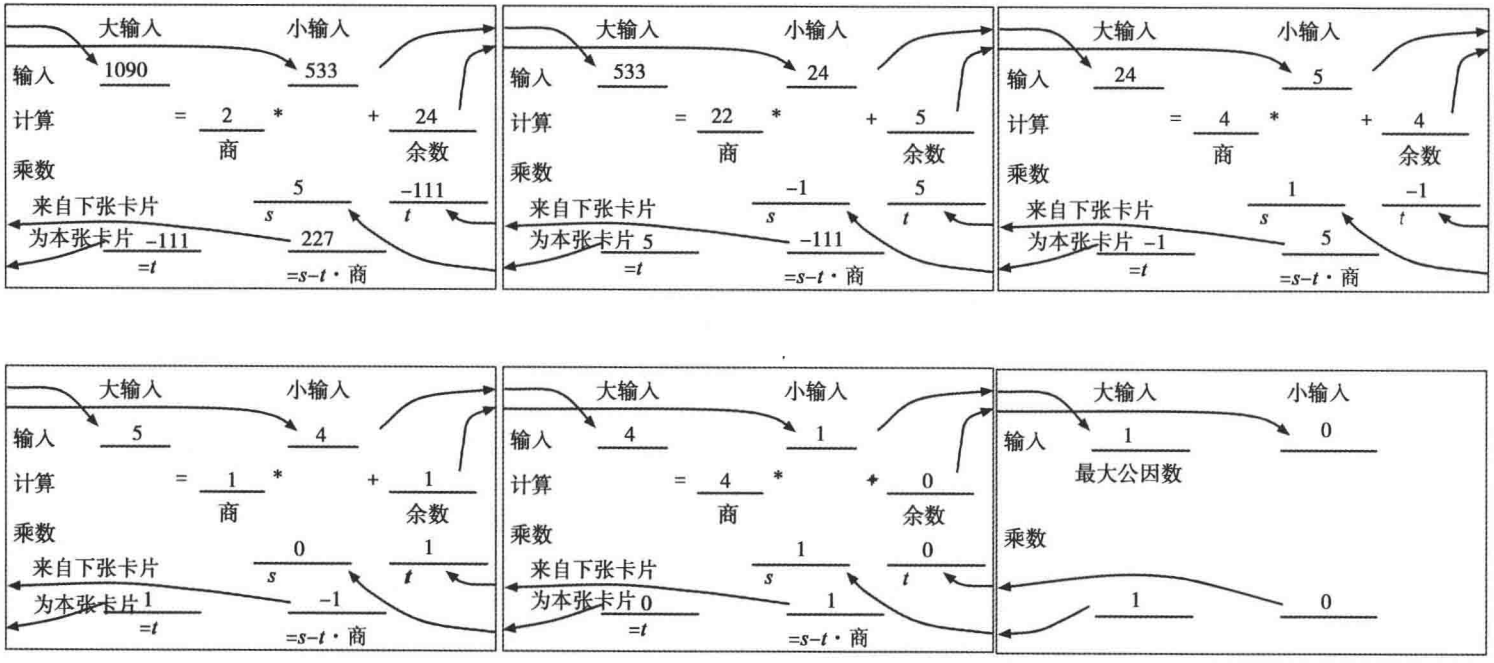
在前向计算的最后一步，当小输入是 0 的时候有一个特殊的卡片。这个特殊卡片提供的乘数分别是 1 和 0。

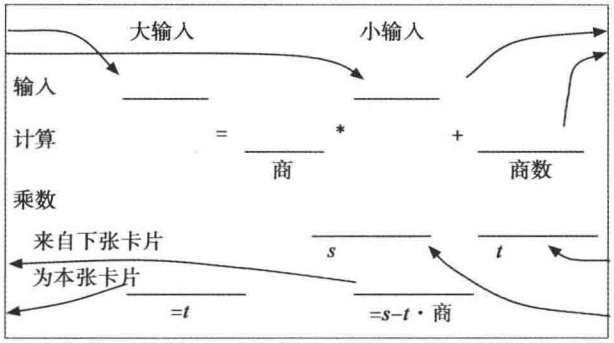


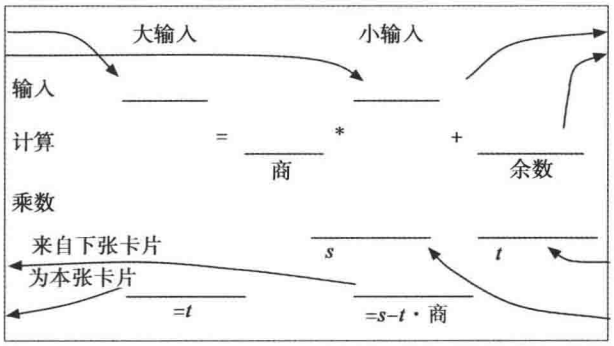
100

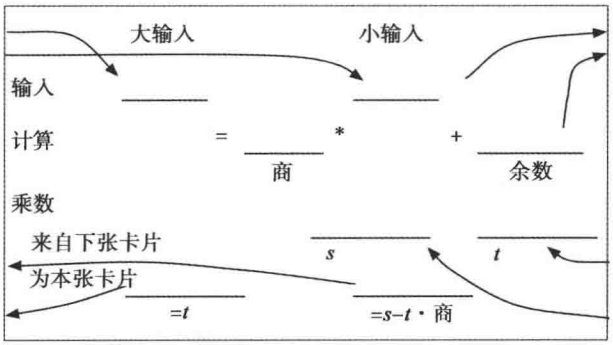
最后一张卡片中的大输入就是两个原始输入的最大公因数。

我们给出一个使用欧几里得卡片完整计算的例子，还有空白的欧几里得卡片以便你复制使用。

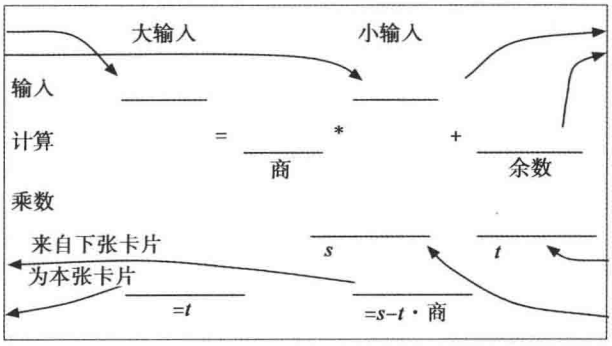

大输入
小输入
输入
1090
533
计算
=
2
*
+
24
商
余数
乘数
来自下张卡片
5
-111
s
t
为本张卡片 -111
227
=t
=s−t・商
大输入
小输入
输入
533
24
计算
=
22
*
+
5
商
余数
乘数
来自下张卡片
-1
5
s
t
为本张卡片 5
-111
=t
=s−t・商
大输入
小输入
输入
24
5
计算
=
4
*
+
4
商
余数
乘数
来自下张卡片
1
-1
s
t
为本张卡片 -1
5
=t
=s−t・商
大输入
小输入
输入
5
4
计算
=
1
*
+
1
商
余数
乘数
来自下张卡片
0
1
s
t
为本张卡片 1
-1
=t
=s−t・商
大输入
小输入
输入
4
1
计算
=
4
*
+
0
商
余数
乘数
来自下张卡片
1
0
s
t
为本张卡片 0
1
=t
=s−t・商
大输入
小输入
输入
1
0
最大公因数
乘数
1
0

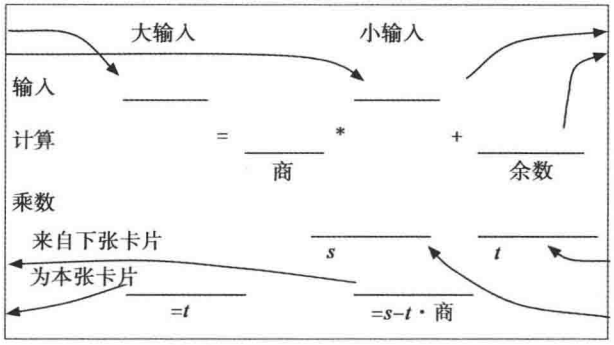

大输入
小输入
输入
计算
=
*
+
商
商数
乘数
来自下张卡片
s
t
为本张卡片
=t
=s−t · 商

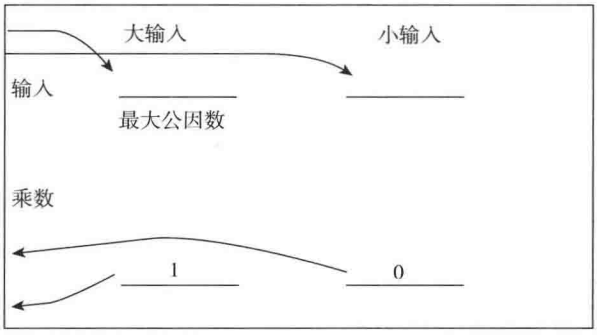

大输入
小输入
输入
计算
=
*
+
商
余数
乘数
来自下张卡片
s
t
为本张卡片
=t
=s−t · 商


大输入
小输入
输入
计算
=
*
+
商
余数
乘数
来自下张卡片
s
t
为本张卡片
=t
=s−t · 商

101
〜
102


大输入
小输入
输入
计算
=
*
+
商
余数
乘数
来自下张卡片
s
t
为本张卡片
=t
=s−t · 商





8.6 欧几里得算法教会我们什么

我们在8.3.1节末尾中看到，一个整数b存在模a的乘法逆元，当且仅当a和b互素。当这个条件成立的时候，更进一步地，欧几里得算法可以为我们计
103 算相应的乘法逆元。

那又怎样呢？让我们回顾第4章的4.3.4节，对于一个模数m，可以通过制作一个模m元素代表元的乘法表找到乘法逆元。既然这样，为什么还需要欧几里得算法呢？

这里有两个原因，一个是计算上的，另一个是数学上的。

- 欧几里得算法的计算需求：当m很大的时候，制作模m乘法表是非常不切实际的。欧几里得算法为计算乘法逆元提供了一种计算捷径。在第13、14章，我们将讨论依赖于计算大模数乘法逆元的密码方案。
- 欧几里得算法的数学需求：模乘法逆元存在性（也就是互素性）的数学判定准则可以帮助我们设计密码方案。在下一章，我们将介绍基于模乘法逆元存在性的密码方案。通过要求模数必须是素数来避免乘法逆元不存在的问题。

8.7 思考题

1. 对下列问题，通过互素性来判断模乘法逆元是否存在。
   (a) 6 关于模数 7 的逆元?
   (b) 14 关于模数 7 的逆元?
   (c) 4 关于模数 25 的逆元?
   (d) 5 关于模数 25 的逆元?
   (e) 21 关于模数 24 的逆元?
2. 回忆第 7 章中欧拉函数内容，拥有模 $\phi(21)$ 乘法逆元的大于 1 的正整数是什么?
3. 考虑整数 18 和 15。
   (a) 两个数的最大公因数是什么?
   (b) 给两个整数 s 和 t 赋值，使得最大公因数等于 $s \cdot 18 + t \cdot 15$。
4. 考虑整数 70 和 40。
   (a) 两个数的最大公因数是什么?
   (b) 给两个整数 s 和 t 赋值，使得最大公因数等于 $s \cdot 70 + t \cdot 40$。
5. 对于输入为 55 和 24 的欧几里得卡片，找到最大公因数以及整数 s、t，使得最大公因数等于 $s \cdot 55 + t \cdot 24$。 104
6. 对于输入为 259 和 105 的欧几里得卡片，找到最大公因数以及整数 s、t，使得最大公因数等于 $s \cdot 259 + t \cdot 105$。
7. 对于输入为 34 和 21 的欧几里得卡片，找到最大公因数以及整数 s、t，使得最大公因数等于 $s \cdot 34 + t \cdot 21$。
8. 使用欧几里得卡片找到 3 关于模数 394820020 的乘法逆元。 105

第9章
A Cryptography Primer: Secrets and Promises

完美保密的某些应用

从第6章可以清楚地知道完美保密在加密中是非常有用的。这个概念在构造其他密码学构建模块中也是很有用的。在本章中，我们讨论两个例子。

9.1 秘密分享与完美保密

完美保密的概念可用于将一个秘密在密码学意义上“拆分”成两部分。每一部分可以给一个不同的人。仅通过自己接收到的部分，每个人对秘密都一无所知，只有两个人一起才可以重构秘密。

试想一下，假如银行行长想将保险柜密码给两个副行长（以防有一天行长不在时，保险柜可以打开），但是想让他们只有联合起来才能使用。他可以使用秘密分享来将密码拆分给两个副行长。

令 $f(clear, key)$ 是一个完美安全密码系统的加密函数。我们将使用这个密码系统拆分秘密。选择何种密码系统不是一个秘密，假设大家都知道选择的密码系统。密码系统的选择限制了被分享的秘密的选择，这些秘密必须是密码系统的可能明文之一。

令 s 为被分享的秘密。在所有可能的密钥中随机均匀选择一个密钥 k。密钥 k 作为提供给第一个人的部分。使用密钥 k 加密 s。密文 $f(s,k)$ 作为提供给第二个人的部分。每一个部分自己不能提供关于秘密的信息：(1) 不管秘密 s 是什么，密钥的概率分布是均匀的，因为密钥就是这样选择的；(2) 不管秘密 s 是什么，密文的概率分布是均匀的，因为密码系统是完美保密的。然而，两个人一起能解开秘密：用密钥解密密文。

例6 令 m 是模数，考虑一个密码系统，其加密函数是

$$f(clear, key) = clear - key \pmod{m}$$

令 s 是0至 $m-1$ 之间的一个秘密数字。对于一个随机密钥 k，秘密的第一部分是 k，第二部分是 $s-k$ 的模 m 代表元。因此两个部分的和模 m 就是秘密。 ■

秘密 s 能在任意数量的人中拆分。假设秘密必须在四个人之间分享。随机选择一个密钥 k_1 并且把它提供给第一个人。令 c_1 为用密钥 k_1 加密 s 得到的密文。现在秘密 c_1 一定要在剩余的三人之间分享。再随机选择一个密钥 k_2 并且提供给第二个人。让 c_2 作为用密钥 k_2 加密 c_1 得到的密文。现在 c_2 必须要在剩余的两人之间分享。随机选择一个密钥 k_3 并且将它提供给第三个人。最终，将通过密钥 k_3 加密密文 c_3 得到的密文 c_4 给第四个人。

例 6a 使用例 6 中的密码系统将秘密 s 在四个人之间拆分，我们最终得到的部分是四个模 m 代表元，它们的和模 m 即为秘密 s。 ■

当然，人们可以分享比加密系统所允许的明文的长度还要大的秘密：简单地把这个秘密用一种普通的方法分成好多分组（比如，第一个分组是秘密的前五个符号，以此类推），然后把每一组秘密分别进行分享。

9.2 门限秘密分享

上述秘密分享的形式是非常基本的。一个在众人中拆分秘密且更有用的方法被称为*门限秘密分享*。我们的银行行长现在有五位副行长，他需要把秘密拆分给他们。然而，她还想允许副行长中任何两个人能重构秘密。在这种情况下，我们称门限为 2。门限秘密分享允许她将秘密拆分给五位副行长，任何人不能单独确定秘密，但是只要有两个人就能准确地重构秘密。在模算术中我们也使用直线方程。在传统算术中，一个直线方程的形式是

$$y = A \cdot x + B$$

107

这就是说，满足该等式的对子 (x,y) 的集合形成一条直线。通常，A 叫作直线的*斜率*（它决定直线的陡峭度），而 B 叫作直线的 *y 轴截距*（它决定直线在 y 轴相交的位置。点 $(0,B)$ 是直线与 y 轴的交点。）

我们将使用相同的想法，但是将它与模算术结合。比如说，我们的模 m 是素数（见第 7 章），其原因将在后面的章节中提到。在模算术中，直线的方程是

$$y \equiv A \cdot x + B \pmod{m}$$

这条“线”由对子 (x, y) 的集合组成，使得 x 和 y 是模 m 的代表元，并且它们满足这个方程。

因为是模算术，这样一条直线看起来很有意思。例如，对于模 $m=17$，

图 9.1 描绘了由下列方程定义的直线：

$$y \equiv 3 \cdot x + 5 \pmod{17}$$

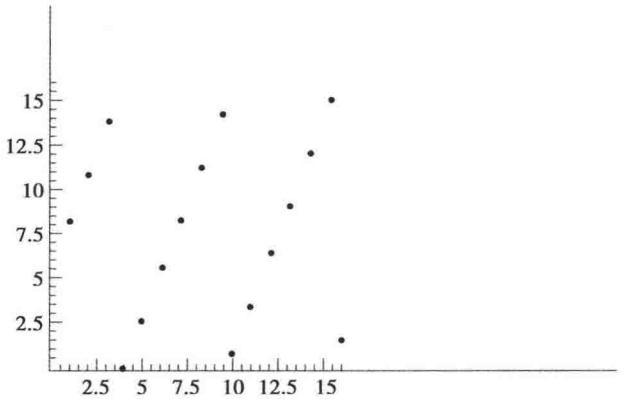


图 9.1 方程为 $y \equiv 3 \cdot x + 5 \pmod{17}$ 的直线

下面讨论如何用模直线分享一个秘密的数字。首先选择一个模数 m，它要足够大，超过所有你可能会共享的秘密值。我们将模数作为系统的一部分，它是公共知识（回想一下 Kerckhoffs 原则）。然后，在模 m 的代表元中秘密地、
108 随机均匀地选择直线的斜率 A。最后，令被分享的秘密数字作为 y 轴截距 B。为了分享秘密，给一号副行长点（1，$A \cdot 1 + B$），给二号副行长点（2，$A \cdot 2 + B$），给三号副行长点（3，$A \cdot 3 + B$），以此类推。

首先我们讨论为什么一个副行长不能单独确定秘密。他唯一得到的信息是一对模 m 的数（X，Y），其中 X 不为 0。他的目标是确定秘密，这个秘密是经过他的点（X，Y）的某条直线的 y 轴截距。然而，有许多直线通过他的点。事实上，对每一个模 m 代表元 β，都有一条穿过（X，Y）的直线，其 y 轴截距是 β。因此一个副行长不能自己排除模 m 任何代表元，基于他所拥有的知识，这些代表元中的任何一个都可能是秘密。

为了证明这一点，我们只需说明存在一条直线穿过（0，β）和（X，Y）两点。你可能记得如何求出经过两个点的直线方程，使用那个公式，会得到方程

$$y = ((Y - \beta)/X)x + \beta \tag{9.1}$$

利用代数知识，可以将两个点代入方程，并验证它们都满足这个方程。因此，确实存在一条直线穿过（X，Y）且 y 轴截距为 β。无论 β 的值是什么，这都是正确的。

让我们用一个非常小的模数 $m=7$ 举个例子。假设秘密值是 4。银行行长从 0，1，2，3，4，5，6 中随机选择一个斜率，假设她选择了 2，则直线方程为

$$y \equiv 2x+4 \pmod 7$$

接下来她计算要分给副行长的点。对于第一个副行长，她计算点（1，2·1+4），也就是（1，6），并把这个点分给第一个副行长；接着她计算第二个副行长的点（2，2·2+4），即（2，1）（记得，我们在做模 7 运算），她将（2，1）给第二个副行长；她继续计算第三个副行长的点（3，2·3+4），即（3，3）；类似地，第四个副行长得到（4，5），第五个副行长得到（5，0）。

假设第四个副行长想试着自己计算出保险柜密码（当然，由于模数很小，他只需要尝试七个可能，但在实际应用中的模数比这个大很多）。他想求出那条通过他的点（4，5）的直线方程，以便计算出 y 轴截距，即秘密的保险柜密码。

“y 轴截距能为 0 吗?”他自己想到。他参照等式（9.1）以得到穿过（0，0）和（4，5）的直线方程，得到 109

$$y = (5/4)x+0$$

因为我们做模 7 的运算，4 的乘法逆元是 2（4·2=1），因此 5/4 的值是 5·2，即 3。因此他得到等式

$$y = 3x+0$$

他自己认为“点（0，0）和（4，5）确实满足该方程，因此那条秘密直线的 y 轴截距可能为 0。”

“y 轴截距能为 1 吗?”他自己想到。再次使用等式 9.1，他获得穿过点（0，1）和（4，5）的直线方程

$$y = (4/4)x+1$$

他把方程重写为

$$y = 1x+1$$

并且意识到，“点（0，1）和（4，5）事实上满足该方程，因此那条秘密直线的 y 轴截距可能为 1”。

“y 轴截距能为 2 吗?”他重复相同的过程并且最终得到方程

$$y = 2x+4$$

（这个方程是真正的秘密直线，但是我们的副行长并不知道。）他得出结论，秘密可能是 2。

“y轴截距能为3吗？能为4吗？能为5吗？能为6吗？”这个副行长依次把这些可能性都考虑了一遍。对于每一种情况，他都设法构造一条合理的直线，穿过他的点（4，5），并且有假定的y轴截距。他相信0，1，2，3，4，5，6都是可能的y轴截距，因此他不能排除任何一种可能性：他自己的点并不能帮助他了解关于秘密的任何信息。

最后，我们说明两个副行长一起可以准确计算出秘密值。两个副行长一起可以提供两个不同的点，因此他们能求出通过这两点的直线方程。一旦有了方程，他们就能确定y轴截距。

例如，假设副行长1和副行长3合法地一起来确定秘密。副行长1有点$(X_1,Y_1)=(1,6)$，副行长3有点$(X_3,Y_3)=(3,3)$，通过这两点的直线斜率为$(Y_1-Y_3)/(X_1-X_3)$，即$(6-3)/(1-3)$。现在$6-3$得3，且$1-3$是5(mod 7)，因此斜率是3/5。5的乘法逆元是3，所以3/5就是$3\cdot 3$，即2。因此他们确定了斜率是2。现在他们知道秘密直线的方程具有形式

110

$$y = 2x + B$$

因为直线穿过他们的点，他们能代入一个他们的点（如点（1，6））得到

$$6 = 2 \cdot 1 + B$$

从这个方程他们可以确定秘密B是4。

我们已经展示了如何在多个人中分享秘密，以使任意两个人都可以重构秘密。此方案中的门限被认为是2。该技术能被一般化，以实现任何需要的门限值。例如，假设有七个副行长，且行长想让他们中的任意五个人能一起确定秘密。她将采取门限为5的方案。这将使用同样的计算，但具体细节超出了本书的范围。

9.3 消息认证码

门限秘密分享中用到的想法可以应用到消息认证码（Message Authentication Code，MAC）中，这是一个检测通过不安全信道发送的消息是否被改变的方法。

假设Alice需要向Bob发送一条消息。他们不关心是否有人读到这个消息（可能他们已经对它加密了），然而，他们想保证Bob收到由Alice发送的准确消息。如果他们通过不安全信道发送信息，则不能保证没有人篡改消息，但是他们能采取措施识别消息是否被改变。

假定 m 是一个素数模，且比任何可能发送的信息都大。假定 m 是系统的一部分且是公开的。此外假设 Alice 和 Bob 之前协商了一个密钥，即一对随机均匀选择的模 m 数 A 和 B。没有其他人知道数字 A 和 B。

现在 Alice 想将消息 X 发送给 Bob，她发送这个消息的同时发送值 $Y=AX+B$（按模 m 计算），Y 就是消息 X 的消息认证码。

消息通过被 Eve 控制的一台电脑，Eve 因此获得消息 X 和相应的消息认证码 Y。在将这些信息发送给 Bob 之前，她可能修改也可能不修改它们。令 X^1 和 Y^1 是 Eve 传送给 Bob 的消息和消息认证码的版本。

Bob 收到两个数字，未定真假的消息 X^1 和未定真假的消息认证码 Y^1。为了 111
确定它们是否被改变，他将检验（X^1，Y^1）是否位于下述直线上：

$$y = Ax + B$$

如果不是，他拒绝它们，因为它们与 Alice 发送的不一致。如果它们在那条直线上，他假定它们未被改变。

Eve 如何骗 Bob 呢？如果她想改变消息的内容（比如她想发送不同于 X 的 X^1），她必须相应地修改消息认证码以使 Bob 接受修改的信息。然而，她不知道修改后的消息 X^1 的消息认证码会是什么。

Eve 所知道的只有点（X，Y）位于那条秘密直线上。为了使 Bob 接受她修改后的消息/消息认证码对（X^1，Y^1），她需要确定这个点也在直线上。正确的 Y^1 值是什么呢？

“值能为 0 吗？”Eve 想。这意味着秘密直线经过点（X，Y）和（X^1，0）。的确有这样一条线，因此 Eve 认为 Y^1 的正确值可以为 0。“值能为 1 吗？”确实有一条直线经过点（X，Y）和（X^1，1），因此，Eve 想，或许 Y^1 的正确值是 1。类似地，任何模 m 代表元都可能是正确值。

事实上，由于 A 和 B 是随机均匀选取的，因而 Eve 需要用到的 Y^1 的值也同样是随机均匀选取的。因此无论 Eve 如何猜测 Y^1，她有 $1/m$ 的可能猜对。对一个较大值 m（比如 10^{20}），这种可能性是小到可以被忽略的。即使尝试百万次，Eve 也不能猜到正确的数字。

9.4 思考题

1. 你公司中的每一个雇员都为登录电脑系统选择一个口令。最近，你的公司决定每个雇员的口令都应该被分享于其他两个雇员，以防万一。每个口令都能

用 0 到 9999 中的数字表示。你的工作是选择秘密分享方案。你可以假设，除非紧急情况下，持有同事口令份额的两个员工不会合谋得到他们同事的口令。你咨询你的三个下属 Larry、Moe 和 Curly。

112

- Larry 说：

  “对每个被分享的秘密口令 s，从 0 到 9999 均匀选择一个随机数字 b。数字 b 是第一个份额。让 $c=s-b \pmod{10000}$，数字 c 是第二个份额。”

- Moe 说：

  “我同意 Larry 的建议，除了一件事情——如果随机数 b 是 0，则份额 c 和秘密是相同的。这非常不安全！因此我建议从 1 到 9999 中均匀选择数字 b，而不是从 0 到 9999 中选择。”

- Curly 说：

  “我同意 Moe 的建议，除了一件事情。如果随机数 b 小于 10，那么份额 c 可能会和秘密 s 有相同的百位数和千位数，这非常不安全！因此我建议数字 b 应该均匀地从 10 到 9999 中选择。”

谁的方案是最安全的，为什么？特别地，考虑一个人收到份额 c 后，因此能获得关于秘密 s 的信息有多少。

2. 为了在紧急情况下 CS007 型保密柜的密码可用，每个助教被分给秘密的一部分。秘密由四个 mod 7 的分组组成。对于每一个数字，Klein 教授选择一个 mod 7 直线，它的斜率是秘密数字，而 y 轴截距是随机选择的。对于每一条线，Klein 教授为每个助教提供一个点 (x, y)。因此，Kevin 从四条直线中每条直线得到一个点（假设是 x 坐标为 1 的点），Mark 从四条直线中每条直线得到一个点（假设是 x 坐标为 2 的点），Sheryl 从四条直线中每条直线得到一个点（点的 x 坐标是 3）。由于安全上的疏忽，你偶然获得了几个 y 坐标，如下表所示。

	第一个分组	第二个分组	第三个分组	第四个分组
Kevin ($x=1$)	4	6		2
Mark ($x=2$)	2		1	
Sheryl ($x=3$)		3	1	

(a) 给出可以从给定的信息中推断出来的每个秘密分组，展示你的工作。

(b) 对于不可以从给定的信息中推断出来的每个秘密分组，告诉我们为什么不能确定，并且告诉我们分组的可能值是多少。

113

3. 你已经看到了门限秘密分享方案：每个应该分享秘密的人在直线上得到一个点，且秘密是直线的 y 轴截距。我们已经将密钥分享在几个人之中，假设你和另一个人打算一起组合出密钥。你的点是 $x=4$，$y=8$，而你的伙伴的点是 $x=5$，$y=0$，模数是 11，秘密是多少？

4. 在本问题中我们要使用 MAC（消息认证码）。这个问题的模数是 11。Alice 和 Bob 预先协商了一个密钥，由两个 mod 11 数 a 和 b 组成。因此 MAC 函数为

$$f(x) = ax + b$$

所以当 Alice 发送一个消息 X 时，她应该在消息中附加消息认证码 $f(X)$。

如果 Alice 和 Bob 加以小心，他们知道仅当密钥只使用一次时，MAC 才是安全的。不幸的是，他们忽略了这一事实，他们发送两个不同的信息及 MAC 码，这两个 MAC 码产生于相同的密钥（相同的一对数 a 和 b）。你（即 Eve）截获了这些消息和消息认证码：

消息：4，MAC：5

消息：1，MAC：9

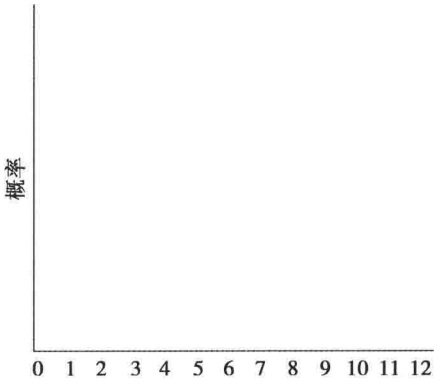
你决定篡改第二条信息，将其改成 3。这条伪造的消息需要伴随什么样的 MAC 才能让 Bob 相信这是合法的？

5. Alice 打算给 Bob 发送一条消息并附带一个消息认证码（生成一个消息认证码的方法就是本文描述的那个）。他们提前协商好一个随机均匀的密钥用于 MAC。消息将被明文发送。可能的消息集合为 0，1，…，12，可能的 MAC 集合也为 0，1，…，12。用模数 13 计算 MAC 的值。

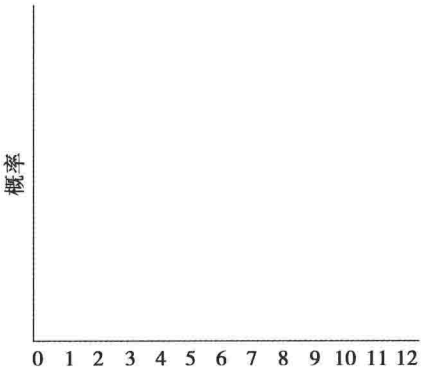
Eve 打算截获消息和消息认证码，并且发送她自己的（假）消息，即 12。她必须选择一个假的消息认证码匹配这条消息，希望愚弄 Bob，使其相信假消息是 Alice 发送给他的。因此她需要知道匹配消息 12 的 MAC 的概率分布（也就是，如果 Alice 发送消息 12，那么什么概率分布的消息认证码将会匹配这条消息呢？）Eve 知道计算 MAC 的密钥是被随机均匀选择的。 114

在下面的每一种场景中，帮助 Eve 找到用于匹配假消息的 MAC 值的概率分布。

(a) Eve 必须在看见真消息和真的消息认证码之前选择她的假的 MAC。给出匹配消息 12 的 MAC 的分布。

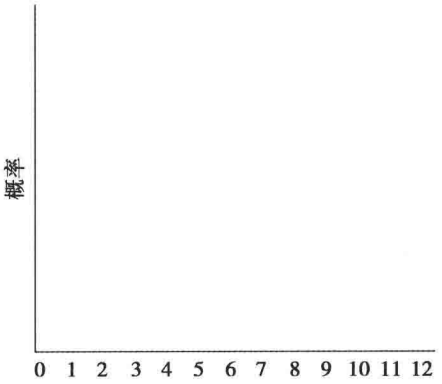


(b) Eve截获真实的消息1以及与之匹配的消息认证码2。给出Eve现在知道的，描述出匹配消息12的MAC的分布。



(c) Alice开始发送消息1和消息认证码2，但是她改变想法，发送消息2和消息认证码4来代替。Eve截获所有这些消息。给出Eve现在知道的，描述出匹配消息12的MAC的分布。

115



6. Klein 教授想要使用秘密分享将 CS007 型保密柜的安全密码分享给助教。这个密码是八位数字。Klein 教授选择 mod 10^8 数字 p，q，r，s，t，使它们满足如下等式。

$$p+q \equiv \text{安全密码} \pmod{10^8}$$

$$r+s+t \equiv q \pmod{10^8}$$

$$r+q \equiv \text{安全密码} \pmod{10^8}$$

（显然，Klein 教授吸入过多粉笔灰，在秘密分享上很迷糊。）他提供 p 给 Kevin，q 给 Mark，r 给 Sandy，s 给 Sheryl，并且，仍然迷糊地把 t 给 Kevin。

对于下面给出的每一组助教，说出他们是否能联合计算出安全密码。

(a) Sandy、Sheryl 和 Mark

(b) Kevin 自己

(c) Sandy 和 Kevin

(d) Sheryl 和 Mark

(e) Sandy 和 Mark

7. (a) 漫步在助教的门外，你注意到在房间的公告牌上有如下信息：

"Kevin：我看了其中一个学生的期中考试成绩。这个学生的分数应该加 20 分。我已经使用一次性密码本（模数为 26）加密了该学生姓名缩写：对于每一分组，

$$cyph = plain + key \pmod{26}$$ 116

密文是 10 12。密钥在便条上，我将从门缝塞进去。Sandy"

你看见在（锁着的）门里的地板上有一张含有密钥的纸条。它是折起来的，所以你不能真正看到密钥。然而，你准备选择你自己的假密钥，将它写在一张相似的纸上，并且将它从门缝儿塞进去，希望 Kevin 看见你的纸条而不是 Sandy 的。

你能造一个密钥使 Kevin 将 20 分加到你的期中考试成绩上吗？如果可以，给出这样一个密钥。如果不能，解释为什么。

(b) 现在你是 Kevin，你去助教的房间并看见如下消息：

Kevin：另一个学生得到了更高的期中考试成绩，使用与前面相同的加密方案——模数 26 的一次性密码本，我已经加密该学生姓名的缩写。密文是 17 23。为了避免上周的惨败，我对每个明文分组使用下列公式计算出了 MAC：

$$f(x) = Ax + B \text{ rem } 26$$

消息认证码的第一个分组是23，消息认证码的第二个分组是7。

你看见写着加密密钥的三个折叠的纸："5 18"、"12 20"及"19 0"。你也能看见一个纸片，写着一对消息认证码的密钥："对于第一个分组，A=1，B=25；对于第二个分组，A=1，B=10。"

117 假设消息认证码密钥是对的，正确的明文是什么？

第 10 章

A Cryptography Primer: Secrets and Promises

计算问题：易解和难解

10.1 计算问题

一个计算问题通过指明输出如何与输入数学相关来定义。这里有一些例子：

立方问题：

- 输入：整数 b
- 输出：整数 $b\times b\times b$

最大公因数问题：

- 输入：整数 a，b
- 输出：a 和 b 的最大公因数

模立方问题：

- 输入：模数 m，代表元 b
- 输出：代表元 c，满足 $c\equiv b\times b\times b \pmod m$

模逆问题：

- 输入：模数 m，代表元 b
- 输出：代表元 c，满足 $c\cdot b\equiv 1 \pmod m$；或者“fail”，如果逆元不存在

模幂运算问题：

- 输入：模数 m，代表元 b，正整数 k
- 输出：代表元 c，满足 $c\equiv \underbrace{b\times b\times\cdots\times b}_{k\text{次}} \pmod m$

计算问题的*实例*是给该问题的输入赋值。

例如：一个模逆问题的实例是“160 模 937 971 的逆”。

10.2 算法

计算问题的*算法*是计算方法，它由一系列确切的步骤组成，接受输入并且产生一个满足这个计算问题中定义的数学关系的输出。

例如，我们已经学习了模逆问题的两个算法。模逆问题就是计算 b 的模 m 乘法逆元。一个算法是写出模 m 代表元的模 m 乘法表，另一个是欧几里得算法。

10.2.1 模幂运算的重复-平方算法

现在考虑模乘幂问题：计算 b^k 的模 m 代表元。朴素的算法是简单地以 b 开始，反复乘以 b，每次都计算出除以 m 的余数。经过 $k-1$ 次乘法运算后，结果就是 $b^k \text{ rem } m$。

重复-平方算法由两部分组成。在第一部分，算法从 b 开始，乘以它自身（对它“平方”）并且取余，然后对结果平方并且取余，接着再对结果平方，以此类推，迭代一定次数。在第二部分，算法将这些结果中的一些合并，把它们乘起来并且取余。为了使算法更精确，需要了解二进制展开的概念。

我们通常在以 10 为基数（“十进制”）的计数法下书写数字。最右的数字是个位，右数第二个数字是十位，右数第三个数字是百位，以此类推。因此 765
119 代表 7 个 10^2，加 6 个 10^1，加 5 个 10^0。

另一个书写数字的体制是以 2 为基数（“二进制”）。二进制中，最右的数字是一位，右数第二个数字是二位，右数第三个数字是四位，右数第四个数字是八位，以此类推。因此 100111 代表 1 个 2^5，加 0 个 2^4，加 0 个 2^3，加 1 个 2^2，加 1 个 2^1，加 1 个 2^0，也就是 39。二进制表示法的数字被称为“位”（bit，binary digit 的简写），按照惯例，最左的数字必须是 1（就像通常我们不把数字 7 写成 007）。

回到重复-平方算法。为了计算 $b^k \text{ rem } m$，我们把 k 写成二进制形式。最左边位的位置告诉我们在算法的第一部分需要进行多少次平方运算。例如，如果最左位在第 2^{11} 的位置，那么需要计算 b^{2^0}，b^{2^1}，b^{2^2}，b^{2^3}，…，$b^{2^{11}}$。因为第一个数字就是 b，所以需要进行 11 次平方运算。

k 的二进制表示中的 1 告诉我们需要合并哪些 b 的幂来得到最终结果。这里有一个乘法计算的例子，需要计算 b^{2184}。对每个乘法行，下表指出了该行被计算的值的表达式。

名称	命令	值
result0	b	b^1
result1	result0 • result0	b^2
result2	result1 • result1	b^4
result3	result2 • result2	b^8
result4	result3 • result3	b^{16}
result5	result4 • result4	b^{32}
result6	result5 • result5	b^{64}
result7	result6 • result6	b^{128}
result8	result7 • result7	b^{256}
result9	result8 • result8	b^{512}
result10	result9 • result9	b^{1024}
result11	result10 • result10	b^{2048}
finalresult	result3 • result7 • result11	b^{2184}

假设指数 k 需要 L 个位表示。那么算法第一部分需要进行 $L-1$ 次乘法运算。在算法的第二部分，需要将在第一部分得到的 L 个结果中的某一些乘起来。因为我们最多需要将 L 个数乘起来，所以在算法第二部分最多需要进行 $L-1$ 次乘法操作。

表示正整数 k 所需要的二进制位数是 1 加上对 $\log_2 k$（以 2 为底 k 的对数）的向下取整。例如，$\log_2 2184 = 11.092\ 757\ \cdots$，这个对数的向下取整值是 11。 120
因此这个公式预测，表示 2184 需要用 12 个二进制位（大多数场合下，忽略“加 1”，而用 $\log_2 k$ 估计二进制位的数目已经足够好了）。

类似地，表示一个数（假设是 m）所需的十进制位数是 1 加上以 10 为底的这个数的对数的向下取整值。例如，$\log_{10} 765 = 2.883\ 661\ 4\ \cdots$，所以向下取整值是 2。因此这个公式预测，表示 765 需要 3 个十进制位。

使用下面的公式实现一个以 10 为底的对数和以 2 为底的对数之间的转换是非常简单的。记住它！

$$\log_2 x \approx 3.3\ \log_{10} x$$

（符号≈表示“近似等于”，3.3 是以 2 为底 10 的对数。）

10.3 预测一个算法需要的计算机执行步数

为了预测一台计算机执行一个算法需要多长时间，需要预测执行单次乘法

需要多长时间。每次乘法的实际执行时间应该依赖于涉及多少位数字。本书中，我们可以说，两个数做乘法需要的计算机的步数是这两个数中较大数的位数（十进制）。

我们在第2章的2.6.2节和2.7.3节中看到，执行一系列乘法运算时，可以把每一个中间结果用代表元替换。采用这种方法，我们能保证只对小于模数的数字做乘法。因此对一个模 m 乘法需要的计算机步数的估计是 m 的十进制表示中的位数。

例如，计算 $b^k \pmod m$ 的朴素算法需要 $k-1$ 次乘法，每次乘法都是模 m 乘法。因为每次乘法需要大概 $\log_{10} m$ 步，所以总的计算机步数大约是 $(\log_{10} m)(k-1)$。

对于同一个计算问题，使用重复-平方算法所需要的乘法次数至多是数字 k 的比特长度减去1所得结果的两倍。因为数字 k 的比特长度至多是1加上以2为底 k 的对数，因此乘法的次数最多也就是 $2\log_2 k$。每一次乘法都要模 m，需要至多 $1+\log_{10} m$ 次计算机步数。因此该算法总的计算机步数大约是 $(\log_{10} m)(2\log_2 k)$。

121

10.4 快速算法和慢速算法：容易问题和困难问题

对一个小的 k 值，朴素算法和重复-平方算法在模乘幂问题的计算机步数上不会有太大差异。然而，当我们增加 k 的值时，它们之间的差异增长极大。一旦 k 是一个100位的数，从前一节中提到的公式中，我们能看出朴素算法的计算步数和重复-平方算法的计算步数有着巨大的差别。前者被称为慢速算法，对于一个100位的输入，它将花费几千年时间去完成计算。重复-平方算法被称为快速算法，即使是1000位的输入，计算机也只需要不到一秒的时间就可以完成。

解决模乘逆元问题的朴素算法涉及构建一个 $m \times m$ 的表，这需要 m^2 次乘法计算。当 m 是7或11时，这并不糟糕。然而当 m 是一个100位的数时，这种算法完全不切实际。相比之下，欧几里得算法要求的卡片数大约是 m 的十进制位数的五倍，并且每一个卡片只涉及一次除法、一次乘法和一次减法。即使是1000位的输入，计算机也会在不到一秒的时间完成欧几里得算法。

通常，快速算法和慢速算法之间的差异随着输入的增大（比如当输入增大到百位或千位时）而变得明显。由于这个原因，用一个公式通过输入长度（比

如，十进制位数）表示算法需要的计算机步数将是非常有用的：它使我们能够预测大输入下算法需要花费的总时间。

这样的预测在实际中不需要完全准确，模乘幂的两种算法之间的差异如此巨大（当输入足够大时），以至于即使每一种算法所要求的计算机步数被低估或者高估一百倍，它们之间的差异也依旧很明显。

快速和慢速算法间的区别导致了容易的计算问题和困难的计算问题间的区别。一个计算问题如果存在快速算法，则称其为容易问题。因此模乘幂被认为是一个容易问题。反之，一个计算问题如果不存在快速算法，则称其为困难问题。注意，在此我们不是要求没有已知的快速算法，而是要求快速算法根本就不存在，这样的计算问题是本质性困难的。

对于很多计算问题，我们不知道它们是否容易：没有人发表快速算法，但也没有人证明不存在快速算法。如果算法研究者没有成功地为一个计算问题找 122
到快速算法，我们经常假设这个问题是困难的。在下一章中会介绍一个假定是困难计算问题的例子。

10.4.1 计算问题和密码学

为什么我们在意一个计算问题是容易的还是困难的？在前面的章节中，我们看到密码方案（比如一次性密码本）如果恰当使用，将不会释放出任何信息。然而，在实际中达到这种安全程度有时是困难的。许多其他的方案都是基于容易问题和困难问题的对立统一的。比如说，我们想要构造一种加密方案，使得一个密钥可执行多次加密，加密和解密对于知道密钥的人来说是容易的，而攻击（没有密钥时解密）则是困难的。

在下一章中，我们将探索这样一些密码方案，它们的安全性依赖于这种容易问题（模乘幂）和（被认为的）困难问题之间的对立统一。

10.5 思考题

1. 假设数字 c 有 n 位（十进制，下同），数字 d 有 m 位。对下列每个问题，利用 m 和 n 给出一个公式。

   (a) $1000 \cdot c$ 大约有多少位？

   (b) $c+d$ 大约有多少位？

   (c) $c \cdot d$ 大约有多少位？

(d) $c \cdot d^2$大约有多少位?

2. DigiComp型计算机在一个时钟周期内可以加两个1位数(并加一个进位,如果需要的话)。因此,为了将两个10位数相加,DigiComp需要10个周期;为了将一个10位数和一个6位数相加,DigiComp需要6或7个周期(取决于最后一次进位)。对于下列每个给定的加法问题,告诉我们DigiComp需要多少时钟周期来解决问题。你的答案不需要精确,允许有微小的偏差。

(a)
```
  12345678
+     1234
```

123

(b)
```
  123456789012345678
+     123456789012345
```

(c)
```
  12345678901234567890123456789012345678901234567890
+12345678901234567890123456789012345678901234567890
```

(d)
```
  123456789
  234567891
+345678912
```

(e)
```
  12345678901234567890
  23456789012345678901
  34567890123456789012
  45678901234567890123
  56789012345678901234
  67890123456789012345
  78901234567890123456
  89012345678901234567
+90123456789012345678
```

3. DigiComp型计算机可以在一个时钟周期内将两个1位数相乘(并加一个进位,如果需要)。因此为了将一个20位数和一个1位数相乘,DigiComp需要20个周期。将一个20位数和一个2位数相乘,比如与37相乘,这个过程包含:(i)将这个20位数乘以7,获得一个20或者21位数;(ii)将这个20位数乘以3,获得一个20或者21位数,并向左移一个位置;(iii)将(i)和(ii)的结果相加。第(i)部分需要20个周期,第(ii)部分需要20个周期,

第（iii）部分大约需要 21 个周期，因此总共大约需要 61 个时钟周期。

对下列每个乘法问题，告诉我们 DigiComp 需要多少时钟周期来解决问题（你的答案不需要精确，有一点偏差没有问题），并做详细说明。**不要害怕走捷径，也就是说，如果你发现了一个模式，尽管使用它。** 124

(a)
$$\begin{array}{r} 12345678 \\ \times\quad 123 \\ \hline \end{array}$$

(b)
$$\begin{array}{r} 12345 \\ \times\ 1234 \\ \hline \end{array}$$

(c)
$$\begin{array}{r} 12345678901234567890123456789012345678901234567890 \\ \times\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad 1234567890 \\ \hline \end{array}$$

4. 假设 DigiComp 需要将两个 k 位数相乘。下列的哪个公式最精确地表达了需要的时钟周期数？参照乘法的过程解释你的答案。注：你选择的公式不需要完全正确，只需要是一个最好的估计值。正确公式的百分率误差对较大的数 k 是较小的；而错误公式的百分率误差对于较大的 k 趋向于变大。可以通过下面的方式检查你的答案：选取两个数，比如两个 20 位的数，利用你选择的公式对这两个数相乘所需时钟周期数进行预测并观察预测的准确程度。就像在前面的问题中一样，不要害怕走捷径：寻找模式。
   - $k+2$
   - $4k$
   - $2k\sqrt{k}$
   - $2k^2$
   - $k^3/8$
   - 2^k
   - 10^k-2
5. 对下列每个公式，想象一台计算机，它计算两个 k 位数相乘所用的时钟周期数由这些公式给出。当 k 取哪些值时，DigiComp 更快？当 k 取哪些值时，想象中的计算机更快？你的答案不需要精确。**提示：你可能发现画函数图有帮助。**

   (a) $10k+4$

(b) $k^2\sqrt{\sqrt{k}}$

(c) $2^k/1100$

125 (d) $2^{\sqrt{k}}$

6. 一个数的以10为底的对数大约是这个数的十进制位数。使用这个事实估计下面给出的以10为底的对数（不要使用计算器）。

(a) $\log_{10}12345678901234567890$

(b) $\log_{10}123456789012345678901234567890123456789012345678901234567890$

(c) $\log_{10}9999999999999$

(d) $\log_{10}1111111111111$

7. 对下列每一个式子，给出 x 的一个值，使得它以10为底的对数近似等于给定的数（有许多正确的答案）。

(a) $\log_{10}x\approx5$

(b) $\log_{10}x\approx7$

(c) $\log_{10}x\approx30$

8. 假设DigiComp需要将数字 a 和 b 相乘。下列哪个公式最精确地估计了执行这个乘法需要的时钟周期？解释你的答案，并将你的答案与 $a=1234567890$，$b=1234567890$ 时的正确答案相比较。

- a/b
- $4a^2b^2$
- $2\log_{10}ab$
- $2(\log_{10}a)(\log_{10}b)$
- 10^a10^b

9. 另一台想象的计算机需要大约 k^3 个时钟周期来将两个 k 位数相乘。下列哪个公式最精确地估计了 a 和 b 相乘需要的时钟周期？解释你的答案，并将你的答案与 $a=1234567890$，$b=1234567890$ 时的正确答案相比较。

- a^3b^3
- $(ab)^3$
- $(10^a10^b)^3$
- $(\log_{10}a)^3(\log_{10}b)^3$
- $\log_{10}(a^3b^3)$

10. (a) 对下列每个以二进制形式给出的数字，给出十进制的表示，说明你是如何获得你的答案的。

    i. 101

    ii. 1111

    iii. 10100111010 126

    (b) 对下列每个以十进制形式给出的数字，给出二进制的表示，说明你是如何获得你的答案的。

    i. 7

    ii. 16

    iii. 632

11. 对下列每个数字 x，估计 $\log_{10} x$。你的答案不需要精确，只要求你的答案是距离正确答案在 1 以内的一个整数。要求不使用计算器。

    (a) 865733500

    (b) 3257100392479038

    (c) 2903483223973759397330024

    (d) 4409566383337997409055252635 63

12. 对下列每个数字 x，估计 $\log_2 x$。再一次，你的答案不需要精确，只要求你的答案是距离正确答案在 1 以内的一个整数。要求不使用计算器。

    (a) 75777（十进制）

    (b) 3337150（十进制）

    (c) 10111011001（二进制）

    (d) 10101010000011111111010（二进制）

13. 我们使用类似下面的表展示计算模乘幂所需要的操作，下面这个表展示的是在某个未指定的模数下计算 b^{2184}，相应地，需要把这里的每个乘法理解为这个模数下的模乘。

名称	命令	计算值的公式
result0	b	b^1
result1	result0 • result0	b^2
result2	result1 • result1	b^4
⋮	⋮	⋮
result12	result11 • result11	b^{2048}
finalresult	result3 • result7 • result12	b^{2184}

对下列每个指数，首先获得指数的二进制表达式，给出你的结果。注意在你

的二进制表达式中有多少位。下一步，准备一张像上面那样的表，指出计算乘幂需要的操作。你的表对二进制指数中的每一位都应该有对应的一行，加上一个额外的行把相应的结果乘起来，以得到预期的最终结果。最后，数出
127 你提出的计算中所需的乘法运算的总数。

(a) b^{16}

(b) b^{128}

(c) b^{257}

(d) b^{194}

14. 对这个问题，你将需要一个计算器。在这个问题中，令两台电脑 MegaComp 和 SlowThink 面对面竞争。我们的“赛道”是计算 $b^{938115730327217}$ 模 3068862310548665。注意，指数和模数都是 15 位数。

MegaComp 是非常快也非常笨的，它的设计者不知道重复-平方方法，所以它使用朴素算法来计算模幂（回顾一下这个算法所需的操作数的公式），然而，MegaComp 一秒执行 100 万步操作。

SlowThink 是非常慢的，它是在早一个时代被设计的，且一秒钟执行 1000 步操作。然而，它使用了重复-平方算法来计算模幂。

在发令枪声中启动你的电脑……

(a) MegaComp 将要用多长时间完成上述模幂运算？提示：你的答案也许最好用年来表示。

128 (b) 对 SlowThink 将要用的时间给出一个好的上界。

第11章

A Cryptography Primer: Secrets and Promises

模乘幂、模对数和单向函数

就像我们在上一章的末尾提及的那样，密码学利用了容易计算问题和困难计算问题的对立统一。我们在上一章看到，下面的计算问题是容易的：即使输入数百位的数字，计算机也能在合理的时间内计算出输出结果。

模乘幂问题

- 输入：模数 m，代表元 b，正整数 k
- 输出：模数 m 的代表元 c，使得

$$c \equiv \underbrace{b \times b \times \cdots \times b}_{k\text{次}} \pmod{m}$$

下面我们将给出一个被认为是困难的计算问题的例子。

模对数问题

- 输入：模数 m，模 m 代表元 b，c
- 输出：非负整数 k，使得 $b^k \equiv c \pmod{m}$，如果这样一个整数存在；否则，输出“不存在”。

如果存在一个整数 k，使得 $b^k \equiv c \pmod{m}$，我们就称 k 是 c 的模 m 对数（以 b 为底）。本书中，我们有时写作如下的数学表达式：

$$k = \bmod m \ \log_b c$$

上述标记法在把 $\bmod m \ \log_b c$ 视为一个函数上具有一定的欺骗性。事实上，正如我们对这个计算问题的描述所表明的，有时对于给定的 m，b 和 c，是没有模对数的。此外，欧拉定理（在数论一章讲述的）也能用于表明，如果存在一个模对数，则必定存在无限个。假设 k 是以 b 为底的 c 的模 m 对数。这意味着

$$b^k \equiv c \pmod{m}$$

由欧拉定理，只要 b、m 互素，

$$b^{\phi(m)} \equiv 1 \pmod{m}$$

两等式相乘，得到

$$b^k b^{\phi(m)} \equiv c \pmod{m}$$

最后，运用乘幂第一定律，得到

$$b^{k+\phi(m)} \equiv c \pmod{m}$$

就是说，$k+\phi(m)$ 也是以 b 为底的 c 的模 m 对数。

因此 mod m $\log_b c$ 在两个方面都满足函数的定义：对于某些输入，它没有解；而对某些输入，它又会有多个解。然而，把它作为一个函数看待通常是很有用的。在这个公式出现的很多情况下，(1) 一开始就知道存在解，(2) 得到哪个解并没有关系，所以我们可以忽略它不构成函数这一情况。

模对数与普通对数间的差异

模对数似乎和普通的非模对数相似。普通的以 b 为底的 c 的对数被定义为实数 x 使得 $b^x=c$。这个等式提示我们使用一个（模）等式来定义模对数。此外，与模对数相似，一个普通对数问题有时也没有解（当输入 c 是非正数时）。你能使用科学计算器计算普通对数，至少是以 10 为底的对数，那么为什么普通对数问题简单到一个计算器就能解决，而模对数问题却会如此困难？

这个答案似乎和近似值有些关系。计算器通常不能计算出一个精确的对数值，只能计算出一个近似的对数值。比如说，计算器告诉我 17 以 10 为底的对数是 1.230 448 9，但是当我计算 10 的 1.230 448 9 次方以验算这个结果时，计算器告诉我计算的结果不是 17 而是 16.999 999。事实上，以 10 为底 17 的对数不能使用有限位数表示出来。那么为什么我会对给出错误答案的计算器感到满意呢？10 的 1.230 448 9 次方非常接近于 17 的事实告诉我们，以 10 为底 17 的对数非常接近 1.230 448 9，因此我的计算器给我的答案非常接近正确答案。另
130 外，因为 10 的 1.230 448 9 次方小于 17，我们可以推算出 1.230 448 9 小于以 10 为底 17 的对数的准确值。这类观察被用于为计算一个好的近似对数设计好的算法。

把这种情形和模算术的情况进行对比。令模数 m 为 97 330 327，那么 88 319 671以 2 为底的对数是多少？也就是说，我们希望计算出一个数字 k，使得

$$2^k \equiv 88\ 319\ 671 \pmod{97\ 330\ 327}$$

通过尝试不同的指数，我偶然得到指数 28 305 819，这导致了一个不同但是近似的数：

$$2^{28\,305\,819} \equiv 88\,032\,151 \pmod{97\,330\,327}$$

这显示 28 305 819 可能是 88 319 671 的对数的一个好的近似。同时，因为 2 的 28 305 819 次方计算结果的代表元小于 88 319 671，我们可以试着猜想 28 305 819 小于 88 319 671 的精确对数。

事实上，真正的对数是 12 314，完全不同于我们猜想的 28 305 819，而是比它小很多。于是我们得出结论，与普通算术截然不同，在模算术中，仅仅因为两个数相近并不能意味着它们的对数相近。

一般而言，一个数的模对数在数值上和该数的普通对数没有相似性。例如，普通的以 2 为底 88 319 671 的对数大约为 26.396 231。一般地，模对数等于普通对数是很少见的（这仅仅当普通对数是精确的并且非常小，以至于这个（普通）对数仍小于模数时才会发生。例如，普通的以 2 为底 16384 的对数是 14，因为 16 384 比模数 93 730 327 小，以 2 为底 16 384 模 97 330 327 的对数也为 14。）

作为单向函数的模乘幂

单向函数是这样一个函数 f，下面的第一个问题是计算“容易”的，而第二个问题是计算“困难”的。

前向：输入：f 定义域中的一个元素 b。

输出：$f(b)$ 的值。

后向：输入：f 值域中的一个元素 c。

输出：f 定义域中的一个元素 b 使得 $f(b)=c$。

因此单向函数是一个很容易从它的定义域元素得到对应的像但却很难从值域元素得到它的原像的函数。 131

认识到一个单向函数并不涉及密钥这很重要。可以假定所有人甚至是潜在的黑客都知道单向函数的所有细节。一个使用单向函数的系统的安全性不依赖于任何它自身的隐秘性，而仅仅取决于后向计算的困难程度。

我们关于单向函数的一个例子是 $f(x)=2^x \pmod m$（译者注：$2^x \pmod m$ 可以理解为 $2^x \text{ rem } m$ 的另一种符号表示，下同，不再一一说明），其中 m 是一个很大的奇数模，比如，有几百位。对于一个输入 b，计算机可以在几十万步内使用重复-平方算法计算出 $y=2^b \pmod m$，这没有问题。然而，由 y 计算出 b 则是求解一个模对数问题。我们相信没有可以解决该问题的好的算法。

当然，这并不意味着每一个后向问题的实例都很难解决。我们在本书中已经看到，例如，如果模数 m 非常小，则找到模对数并不是很困难的。即使模数 m 很大（它应该有的那么大，比如几百位），如果 b 的值非常小，那么 $2^b \pmod m$ 和 2^b 是一样的，这就像我们在普通运算中做幂乘，这是由于指数对于模数来说不够大，所以本质上对运算不会有任何影响。对于这样一个实例，后向计算涉及的仅是一个普通的（精确）对数计算。这种明显的安全缺陷可以归为一个更一般的关于单向函数的使用问题。这个更一般的问题出现在下面要讨论的两个示例应用中。

对某些应用而言，f 是单向的是不够的。在下面第二个应用的讨论中，我们可以看到模乘幂至少有一个弱点使得它并不完全适合于那个应用，即存在一种安全缺陷。本书中，我们忽略这种缺陷。使用模乘幂作为一个单向函数也有可用性缺陷：即使我们有模乘幂的快速算法，即*重复-平方算法*，对于一些应用来说，也还是不够快的。基于这两点原因，在实践中，用其他更复杂的函数作为单向函数。

安全参数

我们已经说过，容易计算问题和困难计算问题之间的差距将随着输入位数的增长变得越来越明显。因此输入越大，安全性就越高。然而，大输入的缺点是它降低了可用性；即使是快速算法，其所需时间也随着输入的增加而增加。

132 为了允许密码方案的用户决定如何平衡这种折中，密码方案需要带有一个所谓的“*安全参数*”。安全参数越高，该方案的安全性就越大，但是对使用该方案的人来说则越不方便。对于指数单向函数，它的安全参数就是模数 m 的十进制位数。

11.1 单向函数在口令安全中的应用

考虑使用口令来限制访问某些资源，比如一台计算机。一个简单直接的方法是在计算机上存储一张大表，叫作*口令文件*，每个用户以及他对应的口令都列举在这个表中（见图 11.1）。当某人试图获得访问权限（通过登录）时，他把用户名和口令提供给计算机，然后计算机会在表中查找该用户名并且把输入的口令与表中该用户名下的口令相比较。如果匹配，则计算机假定登录的人是合法用户并提供了他们真实的姓名，授予该用户访问权限。

用户	口令
Aaron	"bubble gum"
Alice	"mosquito"
Anselm	"cat"
…	…
Zachary	"mosquito"

图 11.1 在一种简单的组织方式中，口令文件的每个条目给出一个用户的口令

在这样一个系统中，口令文件是系统安全中的一个薄弱环节。设法访问了口令文件的黑客立即拥有了所有用户的系统口令，他可以模拟任何用户进入系统。由于人们经常使用相同的口令进入不同的系统，这有时甚至帮助黑客闯入其他的计算机系统，等等。这是 1988 年 11 月的网络蠕虫病毒（一个复制自己并通过互联网传播的无赖程序）使用过的一个伎俩。

为使口令文件对黑客来说不那么有用，单向函数被运用到其中。对应于每个用户名，表格中不直接存储相应的口令，取而代之的是存储口令在单向函数作用下的像（见图 11.2）。现在，当用户试图登录时，会向系统提供自己的用户名和口令，系统将单向函数作用于口令获得其相应的像，然后把这个单向函数的像同口令文件中用户条目里对应的值相比较。如果匹配，则用户被授予访问权限。 133

用户	f（口令）
Aaron	29510086573
Alice	7792389588043
Anselm	193837727847
…	…
Zachary	7792389588043

图 11.2 在一个更安全的系统中，口令文件的每个条目对应了用户口令在某个给定单向函数 f 作用下的像。在这个例子中，f("bubble gum")=29510086573，f("mosquito")=7792389588043。请注意，相同的值被存储在 Alice 和 Zachary 的名字后面，因为他们碰巧选用了同样的口令

那么，这种使用单向函数的方法是如何增强安全性的呢？一个访问了口令文件的黑客不能因此获得每个用户的口令，至少不能直接获得口令。单向函数的定义（也就是"后向"计算是困难的）意味着黑客从存储在口令文件中的值推断出用户的口令是很困难的。

11.1.1 针对使用单向函数的口令文件的字典攻击

口令文件使用单向函数并不能消除黑客操作口令文件带来的风险。为了便于

记忆，用户倾向于选择常用的单词或者名字作为口令。黑客可以准备一万左右的常用口令列表（也就是一个字典），然后将单向函数作用于每个口令，观察得到的结果是否出现在口令文件中。每一个匹配都给出一个用户的口令。

有人可能会说，黑客也可以简单地尝试利用字典里的每一单词登录（这是电影《战争游戏》中出现过的黑客攻击的场景）。毕竟，这种攻击并不需要黑客去访问口令文件。有三个原因使这种攻击相比于首先获得口令文件的攻击方法要低效很多。首先，良好的计算机系统在登录过程中包含了一个延迟：用户被迫在连续尝试登录间隔中等待几秒，因此尝试数千个不同口令是不可行的。相比而言，黑客在口令文件中每秒能测试数万个潜在口令。其次，在一些用户
134 多次尝试登录操作失败后，系统管理员会被提醒有试图攻击的行为正在进行。最后，这种攻击要求黑客独立地为每一个用户名尝试所有口令。根据我们目前所看到的，能够访问口令文件的黑客需要为每一个潜在口令使用一次单向函数，然后快速地查看该单向函数值是否出现在口令文件中。

11.1.2 为口令文件“掺盐”

最后一个关于对口令文件进行字典攻击的观察提示了一种阻止这类黑客攻击的技术。这种技术的目标在于确保口令文件中的每个条目都是用户特有的。当一个用户的口令被输入到系统时，一个额外的数字（称作盐（salt））被选取，盐值可能取决于用户的名字，或者是一天中口令输入的时间——任何因用户不同而可能不同的值。口令和盐值的组合作为单向函数的输入，像通常那样，单向函数的相应输出被记录到口令文件对应的用户名下，同时保存的还有盐值（见图 11.3）。

用户	盐	f（盐·口令）
Aaron	573951212	998778673
Alice	8200294838	485823992
Anselm	784389387	2948872387
⋮	⋮	⋮
Zachary	84390239854	686723745

图 11.3 在一个相对安全的系统，为每一个用户选择一个盐值，单向函数作用于盐值和口令的组合。注意到对于 Alice 和 Zachary 来说，尽管他们拥有相同的口令，但是单向函数的输出却是不一样的

在这之后，当用户试图登录系统并输入用户名和口令时，系统会在口令文件中查找用户名并确定对应的盐值，然后将盐值与用户提供的口令相结合，并作为单向函数的输入。系统计算出结果，并与存储在口令文件中的值相比对。

11.2 单向函数在登录中的应用：s/key

正如在第 1 章所讨论的那样，在罗德岛上，我可以通过在网络上发送我的口令远程登录位于加利福尼亚的计算机。这种方法显然是不安全的：窃听者只要控制一台中间计算机就可以记录用户口令和计算机账户的名称。而一个解决这种安全性问题的早期产品就是 s/key。 135

s/key 系统需要使用一个单向函数 f。为了使用 s/key，首先选择一个密钥，假设它为一个 20 位的数字 s；然后用 s/key 计算机程序去找到 s 在单向函数 f 下的像，然后是像的像，然后是像的像的像，以此类推，如此重复操作大约 100 次。结果可以写成：

$$\underbrace{f(f(f(\cdots(f(x))\cdots)))}_{100\text{次}}$$

我们把它缩写为 $f^{100}(s)$。

接着我们会把这个值提供给加利福尼亚的计算机，它会为我以后的登录存储这个值。

之后，当我想登录加利福尼亚的这台计算机的时候，它会要求我提供为我存储的值的原像，注意，$f^{100}(s)$ 的原像是 $f^{99}(s)$，因为

$$f^{100}(s) = f(f^{99}(s))$$

因为我知道 s 的值，所以能够通过运行 s/key 程序重构出 $f^{99}(s)$ 的值，并将它发送给加利福尼亚的计算机。加利福尼亚的计算机检查我发送的值是否真的是它存储的值的原像。计算机是如何做到这一点的呢？它无法计算出存储的值的原像，因为 f 是个单向函数，这将花费过长而无法承受的时间。取而代之的是，它求出我发送的值的像，并将它和存储的值进行比较。一旦它验证这两个值匹配，它就会存储我发送的新值（代替原来的值），用来进行下一次登录验证。

下一次我试图登录加利福尼亚的计算机时，它将会要求我发送 $f^{99}(s)$ 的原像，因此我发送给它 $f^{98}(s)$。由此可见，在我被迫选择一个新密钥 s 重新初始化系统之前，我能够登录 100 次。

该系统比上述所说的需要通过网络发送口令的系统要更加安全：在窃听者得到我对加利福尼亚的计算机的回应时，我的回应早已到达并且成功登录。我回应的值对于登录不再有用，在下一次登录时，计算机在授权访问之前将会要求不同的回应值，具体来说，是窃听者上次看到的值的原像。窃听者不能计算

出原像，因为函数是单向的。

注意，位于加利福尼亚的计算机并不存储任何秘密信息；我保留着密钥 s 的唯一副本。这样就保证了即使有特权访问加利福尼亚的计算机上的文件，也
136 无法作为我而登录。

11.3 单向函数在承诺中的应用/误用

我将一个秘密写在一张纸上，把它封入信封，并把信封放在你眼前。在这一时刻：

隐藏性：你还不知道我在纸上写了什么。

绑定性：我无法在你不察觉的情况下改变信封内我已经写下的内容。

之后的某个时刻，我把信封交给你，你便能得到我的秘密。

上述描写的场景就是承诺协议的模型，它是密码应用中的一个基本元素，令人惊讶地出现在各种各样的场合中。该协议的第一阶段是*承诺*阶段，在第二阶段，我使你知道秘密，称为*解承诺*阶段（这是一个糟糕的术语）。在第一和第二阶段之间，通常你做出了一些决定，并且向我声明了你的决定。承诺协议的*隐藏*属性确保了你无法基于我的秘密做出决策；而*绑定*属性则确保了我无法根据你声明的决定更改我的秘密信息。

如何实现数字化的承诺呢？也就是说，如果我和你分别在地球的一端，且被互联网联系在一起，如何实现与上述相同的功能？也许最容易想到的办法就是加密。对于承诺阶段，我选择一个随机密钥，用它加密我的秘密，然后发送给你密文；对于解承诺阶段，我发送给你密钥，允许你去解密密文。

利用这种方法实现承诺的问题是至少有一些加密方案是不绑定的。也就是说，它不能可靠地阻止我在得到你的决定后改变我的秘密。原因在于，密钥的可选择性让我对你解密时得到的内容有了一点额外的控制权。我能够根据你的决定选择给你什么密钥。这个问题在加密方案是一次性密码本方案时更加麻烦：即使在我发送给你密文之后，我也没有绑定任意一个特定的秘密，这是由于通过合适地选取我提供给你的密钥，我可以使密文解密成任何我想要的明文。

这个例子表明，当我们使用加密来实现承诺时，承诺的两个目标（隐藏性和绑定性）之间存在冲突。完美加密方案的使用只能确保隐藏性而无法保证绑
137 定性。然而，存在加密方案能够同时实现两种属性（与另外的技巧结合使用）。

在本章中，我们的解决方案将不使用密钥，从而避免通过选择密钥给我带

来额外的自由度。一个单向函数是一种无须密钥而能够实现隐藏的方法。一个基本的（正如我们将要看到的，存在缺陷的）利用单向函数实现承诺的方案如下所示（记住，我们对某些协议提前达成一致，包括单向函数的选择）。

- 为了对我的秘密 s 承诺，我计算 $c=f(s)$，并且发送 c 给你。
- 为了解承诺，我简单地发送给你我的秘密 s。你通过计算 $f(s)$ 并验证它和我在承诺阶段发送的 c 相匹配来确定我没有说谎。

这种实现似乎实现了承诺的两个属性。因为 f 是一个单向函数，知道 c 不能使你确定 s。因为我早已经把值 $f(s)$ 发送给你，我不能在之后改主意，并发给你一个不同于 s 的所谓秘密。

事实上，这些推理都是错误的，在某些特定条件下，上述协议既不隐藏也不绑定。然而，通过一些轻微的变化便能实现这两个属性。

11.3.1 不隐藏

上述简单直接的实现不具备隐藏性的原因与字典攻击有关，这种攻击我们在口令安全中讨论过。可能的秘密集合经常是一个相对较小的集合。只要你从我这接收到 $c=f(s)$，就能尝试将 f 作用于所有可能的秘密上，并检查相应的输出是否与 c 匹配。通过这种方法，在解承诺阶段之前，你能够获得关于秘密的某些信息。

例如，如果你事先知道我的密钥是 0 或者 1。这是在应用中一个常见的案例，被称为“比特承诺”，因为我承诺的是一个单个的比特。如果我已经发送给你 $c=f(s)$，你就能简单地计算 $f(0)$ 和 $f(1)$，看看哪一个为 c 来确定我的秘密 s。

更一般地，像在口令的情况一样，即使可能秘密的范围很大，你也可能对我如何选择秘密具有某些知识。如果存在一个可能的秘密小子集，你怀疑我可能会在这个小子集中选择秘密，你就可以尝试每一个可能的候选值。当然，你的怀疑或许是错误的：我可能没有选择这个集合中的任何一个。然而，如果存在一个不可忽略的机会表明你的怀疑是准确的，我们就应该认为安全性受到了削弱。 138

例如，考虑一个猜数游戏。假定数的范围限定为 1 到 10^{10}。我对我猜的数进行承诺，然后你揭露你的数字，接着我展示我的猜测。如果我的猜测与你的数字相差 10^7 以内，那么我赢。你能在我说出我猜的数字前知道我的猜测么？原则上来说，当你得到承诺 $c=f$（我的猜测）时，你能够把所有 10^{10} 个可能数

字输入 f 来计算每个结果并和 c 比较。你可能并不情愿这样做。然而，如果你知道我的偏好是以朋友的生日作为猜测，你就可能使用极小的工作量命中我的猜测。在某些应用中，一次合理的尝试也许能带来本质性的优势。

在某种意义上，问题出在我的秘密没有足够的不可预测性。为了弥补这个问题，协议中需要更多的随机性。下面是改进的协议。

- 我选择一个大随机数 r（称作一个*临时数*，nonce）仅仅用于这一个场合。我将 r 和我的秘密 s 结合起来，获得一个新的数 k，使得任何知道 k 的人可以计算出 s。
- 为了承诺我的秘密，我计算 $c=f(k)$，并把结果 c 发送给你。
- 为了解承诺，我把数字 k 发送给你。你可以从 k 中推导出我的秘密 s。你也能通过计算 $f(k)$ 并检查它和我在承诺阶段发给你的 c 值相匹配来判断我没有说谎。

在我解承诺前，你关于我选择的临时数一无所知。因为临时数有这么多的可能性，所以组合的 k 值也有很多可能性，你不可能尝试所有的可能：这将会花费你太长时间。因此，这个改进后的协议似乎是安全的，能够抵抗我们考虑的攻击。

用什么公式将数字 s 和 k 相结合？结合的精确方式并没有那么重要，只要双方（我和你）事先商定好就行，因此一旦你知道 k 后就能清楚地知道我的秘密 s。在本书中，我们给出一个简单、合理、有效的结合方法。假设秘密 s 是比 T 小的非负数（在比特承诺例子中，$T=2$，在猜数字例子中，$T=10^{10}+1$）。T 的值是事先约定好的，它是协议的一部分并且被所有参与方知道。一个结合临时数 r 和秘密值 s 的公式如下：

139

$$k = r \cdot T + s$$

由初等数论，因为 s 是非负数并且小于 T，所以 k 被 T 除的余数是 s，也就是 s 是 k 模 T 的代表元。

11.3.2 不绑定

我们实现的承诺协议不是绑定的，原因是单向函数不需要是一对一的。例如，假设单向函数是 $f(x)=2^x \pmod{660\ 761}$，因为模数是 719 和 919 的乘积，于是 $\phi(660\ 761)=718\times 918$，即 502 456。我发送给你 $c=120\ 041$。当到了解承诺阶段，我可以发送 666 或是 666+659 124 作为我的秘密，因为这些值对于

f 有相同的像。

有人可能认为，可以通过限制 f 的输入小于 $\phi(660\ 761)$ 来补救这个问题。这没有解决这个问题，可以验证，$2^{54\,927} \equiv 1 \pmod{660\ 761}$。因此，例如，$f(666) = f(666+54\ 927)$。

可以看到单向性不足以实现承诺。在 14 章中，我们定义了*消息摘要函数*。我们将会看到一个消息摘要函数恰好具有实现承诺所需的属性。

11.4 思考题

1. 考虑下列模乘幂的表格：

模 7 乘幂（底为 3）

x	3^x
0	1
1	3
2	2
3	6
4	4
5	5

这样一个表格能够用于确定模对数（也被认为是离散对数）。对于给定的数 y，$\bmod 7\ \log_3 y$ 的值是最小的非负整数 x，使得 $3^x \equiv y \pmod 7$。例如，$\bmod 7\ \log_3 6$ 的值为 3。（只需要在右边一栏找到 6，然后查看它的左边寻找到 3。）

对于这个问题，构建以 3 为底模 11 的类似表格，使用它来确定下列模对数。 140

(a) $\bmod 11\ \log_3 6 = ?$

(b) $\bmod 11\ \log_3 2 = ?$

(c) $\bmod 11\ \log_3 4 = ?$

2. 前一个问题向你提示了一个计算以 b 为底 $y \bmod m$ 的对数的算法，即基于值 m 和 b 构建一个乘幂表格，然后在第二列中查找 y，并输出对应的指数 x。

(a) 运用欧拉定理（7.7 节）解释为什么表格需要不超过 $m-1$ 行？

(b) 构建一个这样的表格需要多少次乘法？给出基于 m 的公式。

(c) 需要多少时钟周期？基于 m 给出估计时钟周期的公式。

(d) 回顾 10.5 节问题 14 中描述的 MegaComp 计算机。当模为 50 位数的时

候，使用这种算法计算模对数，MegaComp 大约需要多长时间？

3. 有一个已知的比建表方法更好的计算模对数的算法（算法很复杂，本书不做深入解释）。当模数为 m 时，需要的时钟周期数通过如下公式来估计（其中 k 是模的位数）：

$$k^2 \cdot 10^{\sqrt{k \cdot (\log_{10} k)}}$$

(a) 当模数 m 是一个 50 位的数时，使用更好的算法，MegaComp 计算机计算模对数需要多长时间？代入上述公式，利用普通计算器计算。

(b) 现在 MegaComp 已经配备了更好的算法以解决模对数问题。这里我们的目标是探索它的极限。多少位的模数能够让 MegaComp 花费超过 1000 年的时间去计算一个模对数问题？（试试 50 位，60 位，70 位，…）

4. 为选择一个好的安全参数，我们必须考虑所使用的方案中算法（例如加密）运行所需要的时间和 Eve 攻破它所需的时间（比如在没有密钥的情况下解密）。当做这种分析时，我们将假设 Eve 使用的是可能的最好的算法，并且能够使用最快的计算机。

对于基于模乘幂的加密方案，使用的安全参数是模数的位数 k。我们已
141 经看到，重复平方算法对模乘幂运算来说是一个很好的算法：所需的时钟周期数大体为 $13 \cdot 2 \cdot k^3$。我们也知道存在计算其逆运算（模对数）的算法，它需要的时钟周期数大约为 $k^2 \cdot 10^{\sqrt{k \cdot (\log_{10} k)}}$。

假设 Alice 的计算机每秒运行 10^8 个时钟周期。虽然这并不慢，但她必须假设 Eve 可能拥有一台每秒能够计算 10^{11} 个时钟周期的机器。假设 Eve 使用上述的模对数算法。那么 Alice 是否能够选择一个安全参数 k，使得她能够在一秒内就加密一个明文分组，同时还能防止 Eve 花费很少时间就能解密它？请给出一个这样的 k 值并加以解释。注：可以用代数知识来获得答案，但在这种情况下，使用计算器尝试不同的 k 值也是可以接受的。

5. Arnold 提出了一个高度安全的新单向函数方案，安全参数 k 是模数的位数。我们不会精确地告诉你这个函数是如何定义的（它是保密的），但是能告诉你计算需要多长时间（关于 k 的函数）。令 $A(x)$ 表示提出的单向函数。

前向：给定输入 x，计算 $A(x)$ 花费 $2 \cdot k^6$ 个时钟周期。

后向：给定输入 y，计算 y 在 A 函数下的原像花费 $10 \cdot k^{10}$ 个时钟周期。

基于你所知道的，你对提出的单向函数有什么看法？对于推荐用于互联网金融交易，它是否足够安全？

6. Dopey、Sneezy 和 Doc 每个人提出了一个单向函数。对于他们的每一个方案，找到一个像（前向）所需的时钟周期数以及找到一个原像（后向）所需的时钟周期数，在下表中以一个关于安全参数 n 的公式给出。基于这个表格，哪个方案是最好的，为什么？

	寻找像需要的时钟周期	寻找原像需要的时钟周期
Dopey	n	n^2
Sneezy	n^2	2^n
Dox	$n-100\ 000$	$n+100\ 000$

142

第 12 章
A Cryptography Primer: Secrets and Promises

Diffie-Hellman 指数密钥协商协议

12.1 动机

传统的、对称密钥密码系统中的一个难点在于如何让双方共享一个任何其他人不知道的密钥。这在双方无法会面仅能通过像因特网这样不安全的信道进行通信的情况下尤为困难。

可以试图通过事先为每个人提供密钥来克服上述困难。然而，这会引入另一个难题。假设有一百万零一个人要加入通信。我们事先无法得知谁想和谁进行保密通信，因此必须给每一个用户一百万个密钥，每一个密钥对应一个我们可能要通信的人。我不可能记住所有的密钥，因此必须把它们存储在电脑上。假设 Eve 窃听了我的通信并且存储了所有的信息，如果她设法闯入我的电脑并得到了我的密钥，她将可以解密所有的信息。

更糟糕的是，假设有另一个人想要加入通信群。为了新加入的人能够和每一个人通信，我们必须给每一个已在通信群中的人提供一个新的密钥。那么如何才能把这些新的密钥安全地传递给这些人呢？（谁能作为分配密钥的“我们”呢？你相信谁能安全地生成和分发密钥呢？）

12.2 背景

指数密钥协商（exponential key agreement）[⊖] 协议提供了你我在不安全信道通信时协商密钥的一个方法。我们仅能在一次通信会话中使用这个密钥，然后丢弃它。因为我们的电脑不会保留密钥，即使 Eve 后来侵入了我们的电脑，她也无法解密我们以前通信的信息。在这里不需要为使用互联网的人分发数以百万计的密钥，任何两个人都能在他们需要的时候协商出一个密钥。

指数密钥协商的安全性部分地依赖于模对数问题的计算困难性。正如前一

⊖ 也被称为指数密钥交换（exponential key exchange）。

章讨论过的，如果这个问题确实是计算困难的，则模乘幂给我们提供了一个“隐藏”秘密数字 x 的好方法，也就是计算一个依赖于 x 的数，但是 x 不容易从这个数中计算出来。我们利用重复平方法计算 $2^x \pmod m$。

本章中，我们使用乘幂的一些其他特殊性质。特别地，我们将使用一个事实，那就是 X 的 Y 次方再做 Z 次方等于 X 的 Z 次方再做 Y 次方。符号表示如下：

$$(X^Y)^Z = X^{YZ} = (X^Z)^Y$$

这个公式即使在模算术中也是成立的。

12.3 协议

假设 Alice 和 Bob 希望选择一个密钥以帮助他们在不安全信道上秘密通信。假定网络中所有的人都知道模 m 的值，Alice 私下选择一个大随机数 A，并用重复平方算法计算 $2^A \pmod m$，我们把这个计算结果叫作 *AlicePart*，因为这是她对选择共享密钥的贡献。她把 *AlicePart* 发送给 Bob。类似地，Bob 也私下选择一个大随机数 B，计算 $2^B \pmod m$，我们称为 *BobPart*，接着，他把 *BobPart* 发送给 Alice。

现在双方都能计算出他们的共享密钥。Alice 私下通过模算术计算 *BobPart* 的 A 次幂得到她的密钥。同样，Bob 也通过模算术计算 *AlicePart* 的 B 次幂得到他的密钥。根据先前描述的乘幂的性质，Alice 计算的密钥和 Bob 计算的密钥是相同的，如下所示（这里都采用模算术）：

$$\text{Alice 的密钥} = BobPart^A = (2^B)^A = (2^A)^B = AlicePart^B = \text{Bob 的密钥}$$ 144

因此，双方现在有了一个相同的密钥，并能保护他们在不安全网络中通信时的隐私，比如利用传统的单密钥密码系统。

12.4 安全

Eve 能从窃听 Alice 和 Bob 间的通信中得到什么呢？她能得到 Alice 用作 *AlicePart* 的数值和 Bob 用作 *BobPart* 的数值。因此，原理上她可以确定密钥值。例如，她计算以 2 为底 *AlicePart* 的模 m 对数 A。然后，她能够像 Alice 一样计算 $BobPart^A$。当然，因为模数很大，这种方法将花费攻击者 Eve 长到不切实际的时间去计算 A（除非她知道一个解决模对数问题的激动人心的有效算法，然而目前没有）。同样，她可以尝试去计算 Bob 的秘密数字 B，但是这可能同样

困难。

Eve是否还有其他的方法？有没有另一种方法，Eve可以从 *AlicePart* 和 *BobPart* 计算出密钥？我们不知道，可能有，但是目前没有已知的更好的方法。

你的隐私依赖于某些问题是计算困难的，这会让你紧张么？也许，解决这些问题本质上是十分困难的，没有人会发现一个好算法去解决它们，因为根本就不存在。当然，有一些计算问题，我们可以用数学方法证明不会存在好的算法。然而，在算法研究领域已经有一些突破，发现了某些曾经认为不存在的算法。

密码学依赖于不确定性似乎是不可避免的。有人说，密码学家很少能踏实地睡觉。然而，你应该记住一个重要但有点难以理解的观点。对于传统加密系统（比如典型的单密钥加密方案），安全性依赖于一个更复杂和混乱的计算问题的难度。你可能会认为这将使得这类问题更加难以解决，但事实上，混乱也许可以掩盖本质上的简单。似非而是的是，模对数问题在数学上的简单性和清晰性使你更有信心，因为如果存在一个根本性的弱点，它将会更加明显。话说回来，谁知道呢？

12.5 中间人攻击

有一种众所周知的攻击称作中间人攻击 Eve可以用它攻破指数密钥协商协
145 议。这种攻击不依赖于Eve能够解决模对数问题，而是依赖于Eve可以使Alice和Bob确信他们在彼此通信，但事实上，他们是在和Eve通信。Alice通过网络发送 *AlicePart* 给Bob，攻击者Eve截取了这个信息，她选择数值 E 并做模算术计算 2^E 作为自己的 *EvePart* 值，并把 *EvePart* 发送给Alice，假装她自己是Bob且发送给Alice的值是 *BobPart*。Alice通过计算 $EvePart^A$ 得到密钥，Eve也能通过 $AlicePart^E$ 计算出相同的密钥。现在，当Alice用她的密钥加密消息并通过网络发送给Bob时，Eve就能截取信息并且解密。

为了保证Alice和Bob不能检测到攻击，Eve需要对Bob做同样的欺骗。假装成Alice，Eve发送 *EvePart* 给Bob，Bob将 *EvePart* 当做是 *AlicePart* 并发送自己的 *BobPart* 值给Alice。Eve截取这个值，同样也得出了Bob将使用的密钥。每次Alice向Bob发送信息，Eve都截取它并且解密（使用她与Alice共享的密钥），阅读明文信息，然后对明文再次加密（使用她和Bob共有的密钥），然后把它发送给Bob。类似地，Bob每次发送信息给Alice，Eve截取它，然后使用

恰当的密钥解密且再加密发送给 Alice。因此，Eve 可以看到每一条发送的信息，且 Alice 和 Bob 完全不知情（Eve 可以用同样的方法改变信息内容。）

这种攻击表明了通信双方安全地识别对方的必要性。现在有很多方法解决这个问题，但现实中也存在一些方法使参与方不能安全地识别对方，也就是，使用了加密技术来使他们自己确信是安全的，但事实上没有做到。

12.6 思考题

1. 你是 GuinEVEre，Alice 和 Bob 使用以 2 为底模 83 721 983 的指数密钥协商。Alice 发送 78 329 101 作为她的部分，Bob 发送 62 974 812 作为他的部分。你窃听到了它们。

   你想得知他们协商一致了什么密钥。幸运的是，巫师 Merlin 是你的伙伴，他能为你计算一个（且仅有一个）模对数。

   (a) 你会要求他为你计算什么模对数？确保给他所有必要的输入。

   (b) 假设他的回答是数字 Z，利用 Z 给出 Alice 和 Bob 的密钥公式。

   (c) Merlin 故意找茬，拒绝回答他能看到的 Alice 和 Bob 彼此发送的任何数的模对数问题。在这种情况下回答 (a) 和 (b)。 146

第13章
A Cryptography Primer: Secrets and Promises

计算安全的单钥密码系统

13.1 现实世界中安全的分组密码

我们已经看到加法密码是不安全的（除非使用一次性密码本）。如果你坚持使用一个密钥并且以加法密码作为分组密码，那么这个密码系统很容易受到明文-密文攻击。其他攻击方式也同样有效。

然而，确实存在许多相对安全的密码系统。其中最著名的是DES（Data Encryption Standard，数据加密标准）。DES于20世纪70年代被美国国家标准局（National Bureau of Standards，NBS）提出，目的是选择一个用于非机密数据的标准密码系统。1974年，为了响应NBS对密码系统的公开征集，IBM提交了一个早在十年前开发的Lucifer密码系统。NBS请求国家安全机构（NSA）帮助评估这个密码系统。NSA对其进行了修改，将密钥长度从112位减少到只有56位（这是一个折中，NSA本来试图将其减少到48位）。这个系统被NBS最终认证批准使用。需要与政府秘密通信的机构被期望使用DES（除非材料是高密级的）。1979年，美国银行家协会推荐使用DES进行加密。因此DES得到了非常广泛的应用。

这里有一点需要注意，这项技术中的56位密钥无法形成足够大的密钥空间。请记住DES是用于银行交易的。可能的密钥数大约是10^{17}。在1997年，为了响应一个（寻找DES密钥的）挑战，一个叫作DESCHALL的团队在挑战公告发布96天后，通过暴力搜索找到了DES密钥。这次搜索使用了几千台电脑，这些电脑的拥有者通过互联网提供服务。在1998年年初，一个相似的挑战在41天内完成。电子前沿基金会（The Electronic Frontier Foundation）是一个非营利组织，它花费了至少250 000美元建立了一个专用机器Deep Crack，Deep Crack能在56小时内找到一个DES密钥。在2005年，美国国家标准与技术局（NIST）（其前身为NBS）撤销了DES标准。

许多DES的变种更加安全。一个名为3DES的系统包含对明文的三次加密，一次接一次，每一次加密都使用不同的密钥。（双重DES并不比普通的

DES更安全，我们不再探究其中的原因。）3DES有时只使用两个密钥，一个用于第一次加密和第三次加密，另一个用于中间的加密。当DES这样使用时，密钥的位数是112。NIST批准3DES可以使用到2030年。

在DES发明以后，一些与政府无关的个人陆续发明了其他一些密码系统。IDEA（国际数据加密算法）是由Lai和Massey发明的，其密钥长度是128位。有一些系统被提出却被证明是不安全的。NSA设计了一个名叫Skipjack的密码系统，该系统被用于政府授权制造商的芯片制造中。Skipjack的设计开始是保密的。因此，试图攻破Skipjack的密码分析者不知道它是如何设计的。鉴于此，一些人对Skipjack的安全性不大信任。Skipjack于1998年被解密。

在2001年，NIST挑选新的加密标准，并指定高级加密标准（AES）作为新的加密标准。它支持128位、192位和256位三种密钥长度。

13.2 密文分组链

在第3章的3.2节中，我们讨论了分组密码的思想：从一个加密有限长度分组的加密系统开始，我们能够构造一个可以加密任意长度消息的密码系统。我们概述了这样的一种方法，称为ECB模式。在3.4节中讨论过ECB模式是不安全的。在本节中，我们概述一个更安全的分组密码的构造方法，叫作密文分组链（Cypher Block Chaining，CBC）。

ECB的问题是每一个明文分组的加密都独立于其他分组。CBC也许是可以想到的最简单的消除独立性的方法，如图13.1所示。第一个密文分组依赖于第一个明文分组。第二个密文分组依赖于第二个明文分组和第一个密文分组（从而间接地依赖于第一个明文分组）。第三个密文分组依赖于第三个明文分组和第二个密文分组（从而间接地依赖于第一个和第二个明文分组）。依次类推。每一个分组加密使用相同的密钥。 148

特别地，为了得到第二个密文分组，先使用某种方式将第一个密文分组与第二个明文分组进行结合。为达到我们的目的，模加法就足够了：两个数相加的和对m取模，m是$10^{\text{一个分组的位数}}$。然后使用分组加密方法加密两数的和[⊖]。

⊖ 在真实的系统中，结合略有不同。每一个分组以2为基来表示（二进制的，数字计算机所使用的数字系统）。两个分组的每一对相应的二进制位相加后对2取模，得到新的二进制位。由此获得的二进制序列用来表示组合的数字，并使用分组加密对其进行加密。相对于我们使用的系统，这种结合方法对计算机而言更简单。

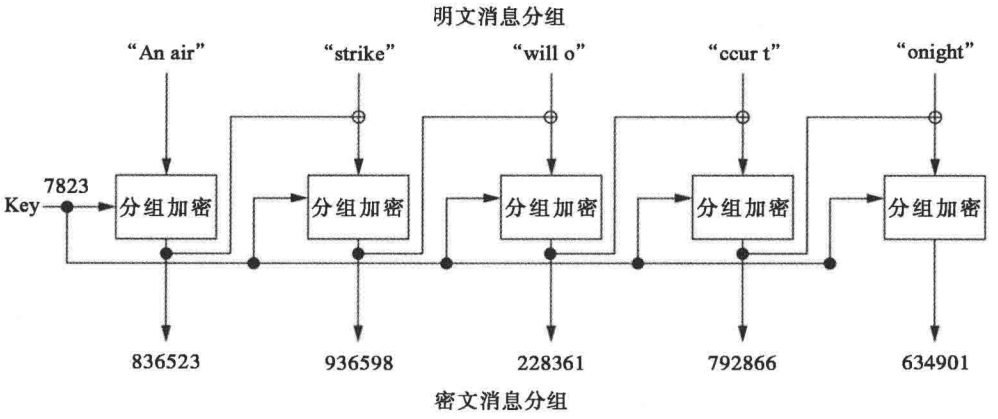


图 13.1 密文分组链图

类似地，将第二个密文分组相加得到第三个密文分组，然后加密该结果。依次类推。

这个方法不像人们希望的那么安全。如果两个消息的前几个分组相同，对应的密文也会相同。基于这个原因，对这个方案进行稍微的改动：在加密第一个明文分组之前，将它与一个叫作*初始向量*（缩写是 IV）的随机数相结合。因此 IV 在密码分组链接中扮演着第 0 个密文分组的角色。如果不知道初始向量，预期的消息接收者将不能解密消息，因此初始向量（明文，也就是说，未加密的）随着加密的信息一起传送。

使用了初始向量后，即使两个明文起始部分是相同的，对应密文的起始部分也很可能是不同的，原因是用于不同加密的初始向量很可能是不同的。实际上，只要初始向量不同，即使明文完全相同，密文也是不同的。使用随机性有
149 助于使系统更加安全。

13.3 指数密码

在本书中，我们只研究一种安全的单钥密码系统，即指数密码。这个密码系统仅以教学为目的。它的速度太慢以至于无法在实际中应用。它对我们而言是有用的，因为它容易描述而且可以很好地说明 RSA（Rivest、Shamir 和 Adleman）的原则。

指数密码系统的实例由模数 m 来确定。你可以把这个模数想象成一个系统参数。这个模数应该选取为一个很大的数。

假设选择了素模数 m 而且每个人都知道 m。加密函数是

$$encrypt(clear, key) = clear^{key} \pmod{m}$$

这里 $clear$ 的值应该是一个正整数且小于模数 m。像往常一样，密钥应是随机选择的。在这种情况下，密钥应该是正数且小于 $m-1$。如果密钥是一个很小的整数（比如 1），加密将是不安全的，但是如果模数是非常大的（理所应当），随机选择的密钥不可能太小。

事实上，不是所有小于 $m-1$ 的正数都能被用来作为密钥，原因将在讨论解密时显现。然而，存在足够的密钥应该不是问题。

在开始加密之前，首先考虑系统是否易受到明文-密文攻击，在这种攻击中 Eve 知道某些明文分组和对应的密文分组。从这些信息中找到密钥相当于求解关于 x 的方程

$$b^x \equiv c \pmod{m}$$

其中 b 是明文，c 是密文。这是一个模对数问题，我们认为它是计算困难的。因此，有理由相信指数密码可以抵抗明文-密文攻击。即使 Eve 知道很多明文-密文对，也可能不会对她有所帮助。

现在考虑解密。这会用到前面章节中讨论过的数论。令 s 为密钥的模 $\phi(m)$ 乘法逆元。那么解密密文 $cyph$ 的方法就是求它的 s 次幂并对 m 取模，即

$$plaintext = cyph^s \pmod{m}$$ 150

这样做可行吗？如果是，为什么？答案部分依赖于 s 是密钥的模 $\phi(m)$ 乘法逆元。乘法逆元的定义是

$$key \cdot s \equiv 1 \pmod{\phi(m)}$$

这意味着密钥乘以 s 减去 $\phi(m)$ 的若干倍数等于 1。令 d 为这个倍数。我们有

$$key \cdot s \equiv 1 + d \cdot \phi(m)$$

现在我们使用欧拉定理来说明解密方法（通常）是可行的。在数论的章节中曾讨论过欧拉定理，现复述如下。

欧拉定理：对于任意模数 m，对任意与 m 互素的数 b，

$$b^{\phi(m)} \equiv 1 \pmod{m}$$

我们想说明，如果首先求明文的 key 次幂并对 m 取模，然后求这个结果的 s 次幂并对 m 取模，则最后的结果就是明文。

$$
\begin{aligned}
(clear^{key})^s &= clear^{key \cdot s} && \text{通过幂运算法则}\\
&= clear^{1+d\cdot\phi(m)} && \text{通过代换}\\
&= clear^1\, clear^{d\cdot\phi(m)} && \text{通过幂指数加法法则}\\
&= clear^1 (clear^{\phi(m)})^d && \text{通过幂指数乘法法则}
\end{aligned}
$$

根据欧拉定理，由于 $clear$ 与 m 互素（由于 m 是素数，每一个小于 m 的正整数都与 m 互素），我们得到

$$clear^{\phi(m)} \equiv 1 \pmod{m}$$

因此得到

$$clear^1 (clear^{\phi(m)})^d \equiv clear^1 (1)^d \pmod{m}$$

当然，$clear^1 1^d$ 是明文。因此 $cyph^s \text{ rem } m$ 就是明文。

现在我们知道解密是如何工作的。它是可行的吗？知道密钥的人可以解密吗？他们需要计算对 $\phi(m)$ 取模，但是这很简单：因为 m 是素数，故 $\phi(m)=m-1$。然后他们需要计算密钥模 $\phi(m)$ 的乘法逆元。现在出现一个问题：仅当密钥与 $\phi(m)$ 互素时，它才会有乘法逆元。如果这样，可以使用欧几里得算法
151 来寻找逆元。因此，如果一个数被选作密钥，欧几里得算法应该被用于寻找模 $\phi(m)$ 的乘法逆元。如果欧几里得算法失败了，应该随机选择一个不同的数作为密钥。因为大多数的数与 $\phi(m)$ 互素，所以只需几次尝试就可以选择一个有逆元的数。

一旦乘法逆元确定了，解密便只需求密文的 s 次幂并对 m 取模。使用重复平方算法能快速实现解密。

13.4 如何寻找大素数

指数密码要求模数是一个大素数。大素数也在 RSA 中使用（将在另一章讨论）。如何获得一个大素数呢？这个问题的答案有两部分。

第一部分是快速检测一个数字是否为素数的方法。在欧拉证明他的定理之前，费马（一个在数学界中“涉猎”的律师和官僚主义者，提出了一些极其重要且有影响力的数学思想）证明了欧拉定理的一种特殊情况，其中模数 m 是素数。在这种情况下，$\phi(m)=m-1$。费马定理是说对于每一个素数 m，对所有与 m 互素的数 b，

$$b^{m-1} \equiv 1 \pmod{m} \qquad (13.1)$$

模数 m 不是素数时上述等式成立吗？一些特殊的合数（非素数）称为卡米切尔（Carmichael）数，它在以下方面的性质与素数相同：如果 m 是一个卡米切尔数，则对于每一个与 m 互素的数 b，等式（13.1）成立。然而卡米切尔数非常稀少。对于大多数合数 m，至少有一半的 b 使等式（13.1）不成立。

这给出一个测试 m 是不是合数的方法。选择一个随机数 b，使用重复平方算法检测等式（13.1）是否成立。如果不成立，则已经表明 m 不是素数（如果 m 是素数，不管 b 的取值是多少，等式都将成立）。如果等式成立，选择另一个随机数 b 并再次尝试。运行这个步骤 30 次左右，如果你发现在每次试验中等式都成立，则可以相当安全地判定 m 要么是一个素数要么是一个卡米切尔数。

我们如何判定 m 是卡米切尔数还是素数？有一种非常相似但稍微复杂的测试可以实现，在此不予讨论，我们也不考虑在上面测试中会出现卡米切尔数的情况。

上面的测试给我们提供了一种可以获得一个 100 位素数的方法。也就是，选择一个 100 位的随机数，并且测试它是否是素数。如果不是，则选择另一个 100 位的随机数再次尝试。继续下去，直到你选择的数被断定为素数。 152

上述过程中需要尝试多少个数呢？答案取决于素数的稀疏程度。如果素数非常少，则在找到一个素数前可能需要做大量尝试。如果素数较为丰富，这个过程就不会花费太长时间。

幸运的是，素数是充分多的。数论中的一个重要定理称为素数定理，它本质上表明在位数为 k 的数中，大约 $2.3k$ 个数中有一个是素数。当 k 相当大时，比如大于 50，这个估计才是比较准确的。对于 $k=100$，这个定理说明大约在 230 个数中有一个是素数。因此，在你碰到一个数是素数之前的平均试验次数是 230 次。这只是平均值——在特定情况下，实验次数可能会更多，但极少会超过太多，比如 2300 次。如果你电脑性能好的话试验速度会足够快。

13.5 思考题

对于前三个问题，你将考虑对应于不同模数的不同的指数密码。对于每一个问题，你将考虑几个不同的密钥。记住对于模数 m 和密钥 k，加密函数是

$$f(clear) = clear^k \pmod{m}$$

也记住解密函数的规则具有以下形式：

$$g(cyph) = cyph^s \pmod m$$

1. 对于下面的模数和密钥值，给出解密的指数 s（也就是说，为了得到明文，密文需要求幂的次数）。如果没有这样的指数存在，解释为什么。

   你会发现如下的模逆表格是有用的。我们把 x 的乘法逆元写成 x^{-1}。

   (a) 密钥=5，模=17

   (b) 密钥=15，模=17

   (c) 密钥=2，模=19

   (d) 密钥=12，模=19

   (e) 密钥=0，模=23

   (f) 密钥=13，模=23

153 在前两个问题中，你将使用欧几里得算法求 s 的值（k 的模 $\phi(m)$ 乘法逆元）。

x	x^{-1} (mod 16)
0	–
1	1
2	–
3	11
4	–
5	13
6	–
7	7
8	–
9	9
10	–
11	3
12	–
13	5
14	–
15	15

x	x^{-1} (mod 18)
0	–
1	1
2	–
3	–
4	–
5	11
6	–
7	13
8	–
9	–
10	–
11	5
12	–
13	7
14	–
15	–
16	–
17	17

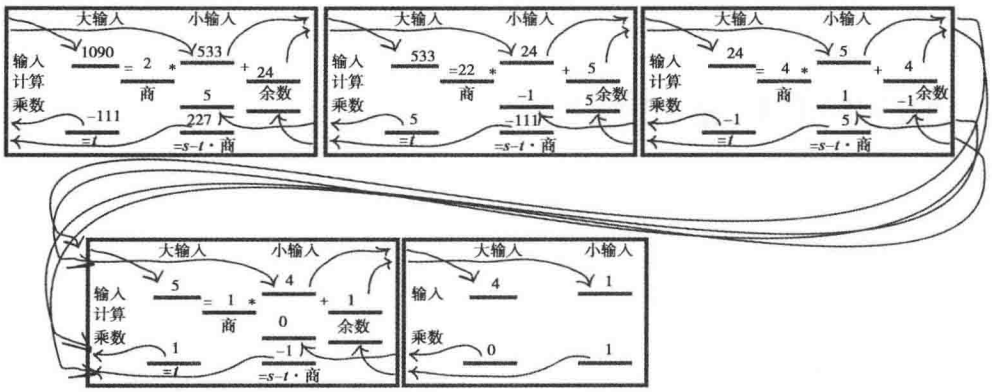
x	x^{-1} (mod 22)
0	–
1	1
2	–
3	15
4	–
5	9
6	–
7	19
8	–
9	5
10	–
11	–
12	–
13	17
14	–
15	3
16	–
17	13
18	–
19	7
20	–
21	21

2. **部分 A**：使用欧几里得卡片确定 s 的值。请拷贝、裁剪卡片，并适当地用胶带将卡片粘在一起。如果给定的密钥 k 没有一个模 $\phi(m)$ 的乘法逆元，你应该指出来，并找出一个大于 1 的整数整除 k 和 $\phi(m)$。

   部分 B：如果你在部分 A 中成功找到一个 s，利用代数、算术和欧拉定理证明你发现的解密函数确实是加密函数的逆运算。

   例：模数 m=1091，密钥 k=533

   部分 A：



部分 B：

令 c 表示明文。

$$\begin{aligned}(c^{533})^{227} &= c^{120\,991}\\ &= c^{1090\cdot 111+1}\\ &= (c^{1090})^{111}c^{1}\\ &\equiv (1)^{111}c^{1} \quad (\text{mod } 1091)\\ &= c\end{aligned}$$

154 ~ 155

(a) 模数 $m=503$，是一个素数。

i. 密钥是 $k=37$

ii. 密钥是 $k=241$

iii. 密钥是 $k=24$

(b) 模数 $m=571$，也是一个素数。

i. 密钥是 $k=133$

ii. 密钥是 $k=77$ 156

第 14 章

A Cryptography Primer: Secrets and Promises

公钥密码系统和数字签名

14.1 公钥密码系统

密钥协商协议为事先互不相识的两方提供了一种协商密钥的方法。然而，有时一方想要在不首先与另一方交互的情况下单方面发送私密消息给另一方。公钥加密于 20 世纪 70 年代由 Diffie 和 Hellman 首次提出，提供了一种解决这个问题的方法。

传统（单钥）密码学使用相同的密钥来加密和解密消息。公钥密码学抛弃了这种方式，允许一个密钥（公钥）用来加密，而另一个密钥（私钥）用来解密。

一个公钥密码体制的建立和使用过程大致如下。每一个准备接收加密消息的人秘密地选择一个私钥并计算相应的公钥。所有公钥是可以公开获得的。如果我想给某人发送加密消息，那么我查找她的公钥，并使用该公钥加密消息发送给她，只有她能够解密该消息。

这种使用不同的密钥进行加密和解密的想法看似简单，但事实上相对于之前的密码学来说是一个惊天动地的突破。很值得思考一下，为什么这个发现这么晚才出现；毕竟，传统的密码学已经使用了几千年。为什么使用两个不同密钥的想法没有更早出现呢？

值得注意的是，私钥和公钥必须紧密联系才能发挥解密的作用。从原则上来讲，它们拥有相同的信息量，这看起来似乎是如果知道了公钥就相当于知道了私钥（类似地，如果你知道一个加密函数，那么别人也会认为你知道了它的逆，即解密函数）。这样来说，拥有两个不同的密钥将会毫无意义——它们只不过是用来表示同样信息的两种不同方式。

虽然这个论点是有说服力的，但是存在缺陷，因为它没有考虑到计算能力的局限性。拥有公钥确实在原则上是能够计算出私钥的。但是，这种计算即使是在运算速度最快的电脑上，也可能需要非常多的时间来完成，以至于这在实

际意义上是不可行的。我猜想，正是这一错误论点的存在使得公钥密码学没能在更早的时间里提出。然而明晰这一缺陷，甚至仅仅是意识到这一缺陷的存在，就已经是非凡的成就了，因为这需要对计算过程有一定程度的敏感与睿智；而在计算机和理论计算机科学发展之前，人们是缺乏这种睿智的。

14.2 El Gamal 密码系统

做了足够的介绍之后，现在我们来描述一个公钥密码系统。这个系统是对 Tahir El Gamal 提出的系统稍作修改的变体。这不是第一个被提出的公钥密码系统——第一个公钥密码系统是 RSA（Rivest，Shamir and Adleman）密码系统——但是只要你了解指数密钥协商协议，理解 El Gamal 密码系统便是非常容易的。现在我们来复习一下指数密钥协商协议。

令 m 作为全系统的模数。在密钥协商协议中，Alice 选择一个秘密随机数 A，然后发送 $2^A \pmod m$ 给 Bob，Bob 同样选择一个秘密随机数 B，然后发送 $2^B \pmod m$ 给 Alice。随后他们都能计算出密钥 $2^{AB} \pmod m$。Alice 计算 Bob 发送给她的 2^B 的 A 次幂来生成这个密钥。Bob 计算 Alice 发送给他的 2^A 的 B 次幂来生成相同的密钥。

El Gamal 方案的基本思想是使用密钥协商协议的一部分，并用它代替初始化过程的部分操作。特别地，在初始化阶段，Bob 选择一个数 *BobSecret* 作为他的私钥，计算 $2^{BobSecret} \pmod m$ 并公布这个数，我们称之为 *BobPublic*。现在如果 Alice 想要给 Bob 发送一个加密消息，她首先查找 Bob 的公钥，发现公钥是 *BobPublic*。接下来，她大体上遵循密钥协商协议，但是她没有等待 Bob 发送给她 *BobPart*，而是使用 *BobPublic* 作为 *BobPart*。也就是说，Alice 选择一个随机数 A，然后计算 $AlicePart = 2^A \pmod m$。然后她计算 *BobPart* 的 A 次幂（在模算术下）作为密钥。她使用一次性密码本的方式，通过将明文与该密钥相加的和对 m 取模来加密明文。最后，她发送给 Bob 两部分信息。一部分是 *AlicePart*，另一部分是密文。 158

Bob 计算 *AlicePart* 的 *BobSecret* 次幂来确定密钥，这本质上与他在指数密钥协商协议中的做法相同。然后他使用这个密钥解密密文并获得明文。

以上的描述可能让人困惑，因为这里有这么多被称为密钥的数。这里的 *BobSecret* 是 Bob 的私钥，*BobPublic* 是 Bob 的公钥。到目前为止我们知道公钥密码系统包含两个密钥，一个是公钥，一个是私钥。Alice 最终用一个密钥来加

密明文；Bob用同样的密钥来解密密文。然而这第三个密钥在此密码系统中的作用是什么呢？这是一个“可丢弃”的密钥，即每个密钥仅使用一次来加密一个消息。虽然它在El Gamal密码系统中的作用非常关键，但是它没有出现在其他公钥密码系统中并且不对公钥密码系统的概念发挥作用。

把公钥密码系统看作是一辆汽车。不管该汽车是使用汽油还是电力驱动，本质上都只有一种驾驶方式。对汽油驱动的汽车来说，化油器是一个重要部分，但是它却是和汽车的概念毫不相关的。类似地，每个公钥密码系统都有一个公钥和一个私钥，从更高的层次来说，它们在公钥密码系统中的使用方式是相同的，只是细节有所差异，从这个意义上来说，这一“可丢弃”的密钥和化油器是相似的。

最著名的（和最早的）的公钥密码系统是RSA密码系统，RSA不包含可丢弃的密钥。实际上，RSA在概念上比El Gamal更简单，但是RSA基于更复杂的数学理论。我们不久将会描述RSA密码系统。

有趣的是，Diffie和Hellman发现了指数密钥协商协议和公钥密码系统的概念，却没有发现与密钥协商如此相似的El Gamal密码系统。其原因可能在于Diffie和Hellman一直在考虑根据陷门单向函数的数学概念来构造公钥密码，并一直在寻找一个基于这个概念的密码系统。El Gamal系统并不是基于这个思想的。鉴于此，或许El Gamal系统并没有进入Diffie和Hellman的视线。我们很快会介绍陷门单向函数的概念。

14.3 关于El Gamal密码系统的更多说明

现在转回来讨论El Gamal密码系统的细节，并作一些技术上的说明。每次一个新消息发送的时候选择一个新的第三密钥是非常重要的（我们从Venona
159 故事中知道重复使用一次性密码本的危险！）。因此，Alice每次想加密一个消息时必须选择一个新的随机数A。如果其他人（比如Bob）想要发送消息给Alice会怎样呢？对于这种情况，Alice拥有她的私钥*AliceScret*和她的公钥*AlicePublic*。Bob选择一个随机数*B*（只是被用来加密一个消息），使用*AlicePublic*作为*Alice-Part*。这里的重点是Alice加密消息时选择的数字A和她用来解密发送给她的消息的私钥*AliceSecret*没有任何关系。

注意到因为每次第三密钥都是通过一个随机过程被重新选择的，所以对同一个明文的两次加密很可能是不同的。也就是说，不像我们之前学习过的所有

其他密码系统，这里的密文不仅仅依赖于明文和密钥，也依赖于一些随机的元素。这可能听起来像是一个缺点（“密文是不可预测的”），但它通常是一个优势：一个窃听者即使窃听了很多密文也不能分辨出它们中的哪一些对应于同一个明文。它存在一个缺点：密文长度至少是明文长度的两倍——它包含两个数，每个都至少和明文一样长——因为密文不仅仅只依赖于明文。

14.4 实践中的公钥密码

当要加密的文档包含比模数更多的位数怎么办呢？注意到这类问题在公钥密码系统和传统密码系统中都出现过。标准的解决方案是将该密码系统用作一个分组密码：把文档分割成小的分组，然后对每个分组分别进行加密。

然而，在实际应用中使用公钥密码系统（El Gamal 或者 RSA）加密大文档会面临一个障碍。为了系统的安全，模数必须非常大（长度至少是几百位）；否则，拥有相应计算能力的窃听者可能破解该系统。然而，由于模数很大，即使是使用重复平方等快速算法，计算乘幂也是非常耗费时间的。这在文档长度很小的时候（几千位）不是一个问题，但是如果文档包含几百万位，这就是一个大问题了。

然而，公钥密码的优点还是非常大的，以至于不能被完全放弃。那么解决方案是什么呢？使用公钥密码加密一个 100 位的数 k 并发送密文。然后用 k 作为某个快速单钥密码的密钥（如 DES）对文档进行加密并发送密文。接收者首先使用它的私钥解密得到密钥 k，然后使用 k 作为 DES 的密钥解密整个大文档。 160

14.5 签名

在使用纸和笔的物理世界里，我通常在文档上签名来表示认可这份文档。在此以后，任何看到这份签名文档的人都确信我认可该文档。例如，如果文档是一个合同，我的签名就是我认可该合同的实体证据。此外，伪造签名被认为是具有一定难度的，别人伪造一个看起来像是我签的签名是很困难的。

在数字世界里，拷贝一份电子文档的一部分到另一份文档是非常容易的，对一个文档进行签名的概念也因此更加微妙。在物理世界里，我们试图让自己所有的签名看起来都是一样的，因为如果我的签名变化很大，一个被伪造的签名很可能会被看作是我的签名。在数字世界里，我的签名不应该简单地由一个

附在文档结尾的、固定的数据片段组成，否则窃听者看到一个合法签名的文档后，就可以将这块数据片段拷贝到别的文档中去，从而成功地伪造了我的签名。

相反，我的数字签名应在数学上依赖于文档的内容。通过验证文档和签名及其之间数学关系的成立，你可以确定一个文档上我声称的签名的确来源于我。因此，为了让其他人能够验证一个签名的确是我的，所要求的数学关系应该是公开的。与此同时，为了满足只有我自己才能产生我自己的签名，这种数学关系应该与某些只有我自己持有的秘密信息紧密相关。

现在回忆一下公钥密码的概念，在公钥密码中，任何人都能对发送给我的消息进行加密，但是只有我能够解密。公钥密码系统和数字签名都依赖于每个人均持有一个公共数据和一个秘密数据。秘密数据是只有本人知道的一个数；知道该秘密的人可以完成某些任务，例如解密或产生签名。某个人的公共数据是一个与其有公开联系的数；给定该人的名字，任何人都可以确定相对应的公共数据，并使用该数据加密发送给他的消息或者验证他的签名。

公钥密码系统和数字签名的密切相似性并非巧合；这两种思想均由 Diffie 和 Hellman 发现并发表于 20 世纪 70 年代的一篇论文中。实际上，他们描述了一个抽象的数学构造，即*陷门单向函数*，它可以作为公钥密码系统和数字签名
161 的基础。

14.6 陷门单向函数及其在公钥加密和数字签名中的应用

我们已经了解了*单向函数*的思想，这种函数正向计算很容易（给定一个输入，计算对应的输出）而逆向（给定一个输出，计算对应的输入）是计算困难的。Diffie 和 Hellman 采用这一概念并对其做了微小却意义重大的修改。如果存在一个秘密信息使得某单向函数的逆向计算变得容易，则称这个单向函数为*陷门单向函数*，而这个秘密就称作*陷门*。

让我们看一下一个陷门单向函数在公钥密码中的应用。为了能让别人给我发送加密消息，我构造并公开一个陷门单向函数，但只有我知道陷门。这个函数被用来加密。由于该函数是公开的，因此任何人都可以给我发送加密消息；函数的输入是明文，输出是密文。解密需要使用该函数的逆，也就是说，从函数的输出（密文）计算出输入（明文）。对于不知道陷门的人来说这个计算是困难的。因为我知道陷门，所以可以容易地进行计算。

接下来，让我们看看一个陷门单向函数如何被运用到数字签名中去。同样，我构造并公开一个陷门单向函数，且只有我知道陷门。为了获得一个文件的签名，我以这个文件作为输出来确定该函数的一个输入，这个输入即为我对该文件的签名。因为我知道陷门，我可以做这样的计算。任何看到该签名的人均可以将它作为输入计算上述函数，并得到一个输出。如果该签名是合法的，这个输出便与文件准确匹配。除了我以外，没有任何人能针对一个给定的文件生成相应的签名，因为除我以外没人掌握该秘密，即陷门，它使计算函数的逆变得很容易。

在本章中，我们描述了一个函数，据我们所知，它是一个陷门单向函数。也就是说，人们相信如果不知道陷门，就没有好的算法能够对该函数进行逆向计算，但是这个观点并没有被证明是正确的。这个函数最早由 Rivest、Shamir 和 Adleman 提出，并用在公钥密码和数字签名中，基于这个原因，该公钥密码系统又被称为 RSA 密码系统，且数字签名方案也被称为 RSA 签名方案。

注意到上述使用陷门单向函数的方法并不是唯一构造公钥密码和数字签名的方法。实际上，El Gamal 公钥密码系统不是基于陷门单向函数而构造的。El Gamal 也提出了一个不基于陷门单向函数的数字签名方案。 162

14.7 RSA 陷门单向函数

下面介绍一种构造陷门单向函数的方法。秘密选择两个大素数（几百位）p 和 q，并且 $p-1$ 和 $q-1$ 不能被 3 整除。然后令 p 和 q 相乘，得到一个数 m。现在公开参数 m，这就是你的公钥。这个单向函数是模 m 三次方：$x \rightarrow x^3 \pmod{m}$。对于任何知道你的公钥的人来说，根据输入计算输出是容易的。然而，任何不知道 m 的素数因子 p 和 q 的人进行逆向计算被认为是困难的。

你知道这些数字，因为它们是由你选择的，因此你可以进行逆向计算。计算 $p-1$ 乘以 $q-1$ 得到 $\Phi(m)$，然后计算 3 关于模 $\Phi(m)$ 的乘法逆元 s。最终，为了计算一个数的三次方根模 m，需要计算这个数的 s 次幂并对 m 取模。

实际上，RSA 并不告诉你使用数字 3；你可以使用另外的数字。例如，只要 $p-1$ 和 $q-1$ 不能被 5 整除，就可以使用数字 5。使用小的指数有一些缺点，但是可以通过使用随机加密来避免（在关于安全的单钥加密的章节中讨论过）。

14.8 RSA公钥密码系统

我们可以在密码系统中使用这个函数。如果你的朋友Alice想要给你发送一个只有你可以读的消息，她把消息取三次方进行加密并把其模 m 代表元发送给你。你可以通过计算这个数的 s 次幂并对 m 取模来进行解密。

14.9 RSA数字签名方案

我们也可以在数字签名方案中使用这个函数。比如说数 b 代表着你要签名的文档。你可以通过计算 b 的 s 次幂并对 m 取模来计算 b 关于模 m 的三次方根。这个结果就是你的签名。你可以把该文档和对应的签名发送给任何拥有你公钥 m 的人，她可以验证该签名是否和文档匹配。也就是说，她计算该签名关于模 m 的三次方并验证是否和文档相匹配。

至于安全性，我们相信（希望）窃听者Eve不能伪造签名，也就是说，她
163 不能对她选择的文档求出关于模 m 的三次方根。即使Eve看到一个文档和你生成的签名，我们相信，对这个文档做轻微修改后，她也不能计算出一个签名。

14.10 消息摘要函数

有很多很多关于数字签名的应用。数字签名在实际应用中可能比加密更为重要。例如，一个软件公司可能需要通过因特网给用户发送一个新的Web浏览器软件。有一些黑客可能在软件传送给用户的途中对软件进行修改，引入一些安全漏洞。为了能够让用户验证收到的软件拷贝是合法的、未被修改的，软件公司将软件看作一个文档，然后计算并公开针对该软件的数字签名。当一个用户收到他的拷贝时，他可以将该数字签名取三次方（对软件公司的公钥取模），然后检测该签名是否和软件相匹配。

然而，正如公钥密码一样，数字签名存在一个实用中的困难。对于比较大的文档（比如说软件），模数也必定会很大，因此计算相应的数字签名将会花费很长的时间。

另一方面，RSA签名方案还存在一个安全性困难。Eve在不知道相应的公开模数的私钥的前提下，可以构造出被认为是合法的［文档，签名］对。要注意，通过这种方式产生的文档是没有意义的。尽管如此，这在很多应用中都是

一个安全漏洞。

解决方案涉及使用密码学哈希函数或者消息摘要函数。这个函数将一个很长的文档作为函数输入然后输出一个很小的数（40 或 50 位）。输入中任何小的改变都能造成输出的巨大变化。此外，给定函数的一个输出，找到能够生成该输出的任何输入都是非常困难的。这么看来，密码学哈希函数与单向函数非常相似。不同的是密码学哈希函数不是双射（一对一）函数，所以没有逆函数。然而，基本的思想是一样的——“前向计算容易，逆向计算困难”。更重要的是，一个密码学哈希函数还需要满足额外的安全性要求，至少具备下述性质之一：

弱无碰撞：给定一个文档 d，计算一个不同的文档 d' 使得 d 和 d' 在哈希函数下的像相同是计算困难的。

强无碰撞：计算两个不同的文档 d_1，d_2，使得它们在哈希函数下的像相同是计算困难的。 164

第二个条件隐含着第一个条件，因此略强。对于某些应用，这个更强的条件是必需的。对于数字签名应用，较弱的条件就足够了。

为了针对长文档构造数字签名，将该文档作为一个公开已知的密码学哈希函数的输入，输出是一个相对小的数。然后使用 RSA 数字签名方案去计算这个小的数的数字签名。由于模数只需是几百位长，因此计算速度会很快。然后发送文档和数字签名。接收者可以按照如下方式验证文档和数字签名的有效性。首先将文档作为密码学哈希函数的输入，以计算输出（正如你做的那样）。然后将数字签名取三次方（对你的公开密钥取模），然后验证是否和这个密码学哈希函数的输出相匹配。

如果密码学哈希函数是弱无碰撞的，黑客就不可能在途中修改文档且不被检测到；对于修改过的文档，该签名将是非法的。

存在可以非常高效地生成输出的密码学哈希函数，即使输入是一个很大的文档。这些函数被认为（期望）是安全的（“逆向计算是困难的”）。然而，其中的某些方案（如 MD4 和 MD5）已经被证实是不安全的。使用者后果自负！

14.11 消息摘要函数在承诺中的应用

回忆第 11 章，我们提出使用单向函数实现承诺。我们展示了使用一个随机数来保证承诺直到恰当的时间才能被打开。然而，我们看到使用单向函数并不

能保证承诺是绑定的。

消息摘要函数提供了一个解决该问题的方法。如果一个单向函数是强无碰撞的（因而是一个消息摘要函数），那么承诺是绑定的。令 f 是一个强无碰撞的消息摘要函数。假设我想对某个值（记为 d_1）通过发送给你 $f(d_1)$ 进行承诺，稍后可能改变我的想法并声称 d_2 是我的原始承诺值。为了让你相信我，d_1 和 d_2 必须在函数 f 下拥有相同的像，即 $f(d_1)=f(d_2)$。因为 f 是强无碰撞的，因而无法找到这样的一对值 d_1 和 d_2。因此阻止了我作弊。

14.12 思考题

1. Bilbo 提出了以下消息摘要函数，它被用作对任意长度的文档进行签名。该
165 函数是 $f(x)=2^x \bmod m$，其中 m 是长度为 100 位的素数。Bilbo 建议将文档表示为一个大的数（可能有上千位），然后计算该数在函数 f 下的像。得到的消息摘要是一个模 m 的代表元。

   现在，我们知道 Bilbo 的函数并不是一个理想的消息摘要函数，因为对于一个长文档，需要几分钟甚至几个小时来寻找相对应的像。如果它是一个安全的消息摘要函数的话，我们或许愿意等待这么长时间。不幸的是，它非常不安全。

   (a) 描述消息摘要函数的安全目标，并解释如果在数字签名方案中使用不安全的消息摘要函数，可能会发生什么不好的事情，为什么会这样。

   (b) 说明 Bilbo 的函数没有实现安全目标，并说明如果它被用在一个签名方案中，Eve 可能利用这个缺陷来修改一个签名的文档，且不被发现。提示：回顾欧拉定理。

2. 你看到 Boris Badinov 正在加密一个消息，并设法读到了开始的几个明文分组："Dear Natasha: I've recruited a new spy in the CIA. His name is"

   你冲向你的窃听装置，并设法窃听所有的密文分组。现在你转向你的超级计算机集群……

   在以下的每一个场景中，简要描述你如何从密文确定其余的明文。特别地，对于每个场景，从下列（i）、（ii）或（iii）中选取：（i）可能成功；（ii）不可能在一年之内成功；（iii）无论你活多久，都不可能成功。

   (a) 每个分组通过明文和密钥 k 相加，然后模 1000000000000 进行加密。所有分组使用同一个密钥。

   - 攻击？

- 成功的可能性?

(b) 同上，每个分组都使用模 1000000000000 加法进行加密，但是现在每个分组选择不同的密钥。

- 攻击?
- 成功的可能性?

(c) 每个分组使用指数密码加密，模为素数 251003。所有分组使用相同的密钥。

- 攻击?
- 成功的可能性?

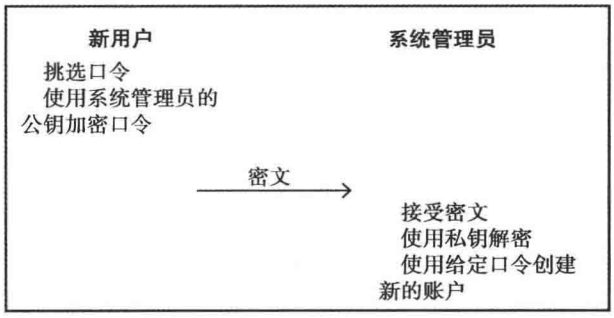
166

(d) 每个分组使用 Natasha 的 RSA 公开模数 6355170039495021706507715-3434685637685852255386187135494209917 … 6495012237225563812-839425115194276145439 和一个公开指数 3 进行加密。

- 攻击?
- 成功的可能性?

3. 使用 RSA 加密最直观的方法是直接使用 RSA 加密明文，也就是说，计算明文的接收者的公开指数次幂（通常是 3），并对接收者的公开模数取模。正如 14.4 节中提到过的，一个更好的方法是选择一个大的随机密钥 K，直接使用 RSA 加密 K 并发送 K 的密文，然后使用单钥密码系统（如 DES 和 RC4）加密真正的明文，并发送生成的密文。

考虑如下的方案：每当一个新用户需要一个账户，她选择她的口令，然后使用系统管理员的公钥对口令进行加密，并通过电子邮件将密文发送给系统管理员。



(a) 说明如果使用直观的 RSA 加密，Eve 可能受益于字典攻击。

(b) 为什么更好的 RSA 加密方法能使字典攻击无效?

4. 就像我们之前使用的基于普通对称密码的挑战-响应协议（在敌我系统中，

识别朋友和敌人），可以使用基于公钥密码（特别是 RSA）的挑战-响应协议。目标是使一方有可能知道她正在和谁通信。假设 Carole 正在和声称自己是 Dave 的人进行通信。Carole 知道 Dave 的公钥（她之前已经得到了 Dave 的
167 证书）。Carole 发送一个随机的挑战 R 给另一方，另一方返回 Dave 对 R 的签名作为回应。Carole 可以使用 Dave 的公钥检查 R 的签名是否有效；因为只有 Dave 拥有私钥，该私钥允许某人像 Dave 一样进行签名，Carole 确信她正在和 Dave 进行通话。

Carole 根据模 m 代表元的均匀分布来选择 R，其中 m 是 Dave 的公开模数。假设 m 是一个大的数（非常大以至于无法分解），因此 R 的选择可能性非常多，以至于 Dave 之前已经计算过特殊选取的 R 的数字签名的可能性非常小。任何人在不知道私钥的情况下能计算 R 的签名也是非常不可能的。

考虑以下能够让双方互相识别对方的协议。A 声称自己是 Alice，B 声称自己是 Bob。

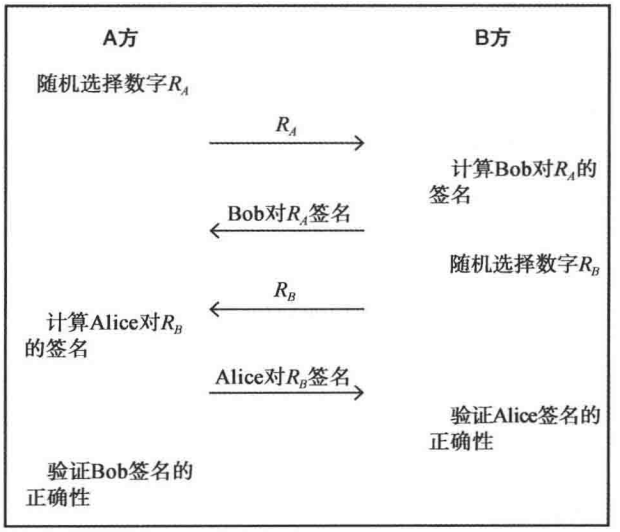
(a) A 选择一个随机挑战 R_A 并发送给 B。

(b) B 发送 Bob 对 R_A 的签名以及他自己的随机挑战 R_B 作为响应。

(c) A 发送 Alice 对 R_B 的签名作为响应。

假设公开模数很大，因此 Eve 很难对其进行分解。

找到一种方法使 Eve 能够让 Bob 和 Alice 相信他们均在与对方通信，实际上他们都在与 Eve 通信。



5. 我们描述一个于 1989 年提出的协议，它能使人们在无线电网络中秘密通信。该协议使用一个可信中心来充当中间人。在这个协议中，我们忽略认证和身份识别：证明你是谁不是必要的。 168

这个协议只要求可信中心拥有一个 RSA 公钥/私钥对。所有的其他参与方都知道可信中心的公钥，但是他们都不知道相应的私钥。

协议运行如下。假设 A 想要和 B 进行秘密通信。

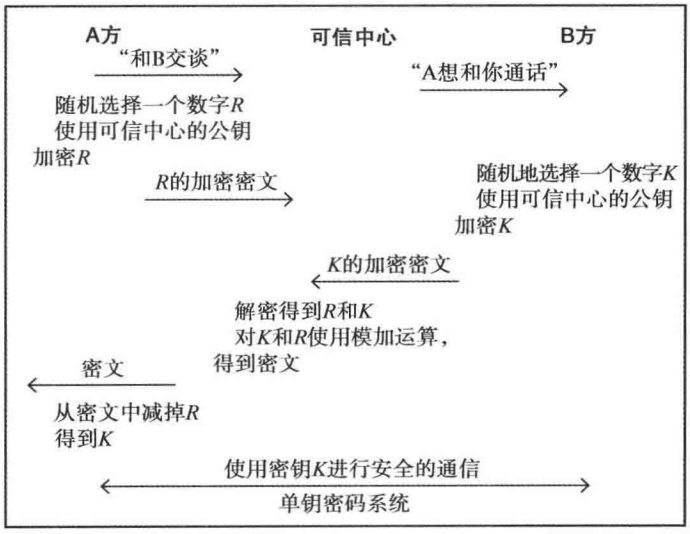
(a) A 选择一个大的随机密钥 R，使用可信中心的公钥对 R 进行加密，然后将密文发送给可信中心。

(b) 可信中心将通信的请求通知 B。

(c) B 选择一个大的随机会话密钥 K，使用可信中心的公钥对 K 进行加密，然后将密文发送给可信中心。

(d) 可信中心解密 A 的密文，获得密钥 R，解密 B 的密文，获得密钥 K。使用一次性密码本用密钥 R 加密 K（使用模算术将 K 和 R 相加），然后将密文发送给 A。

(e) A 解密密文，获得 K。现在 A 和 B 拥有共同的密钥 K，因此他们可以使用普通的（单钥）密码系统加密发送给对方的所有消息。



假设可信中心的行为是诚实的。这个协议看起来很安全，但是有一个相当明显的缺陷。两方 C 和 D 可以合作并窃听所有的通信，以此来解密 A 和 B 之间的通信。 169

你的任务是找到实现的方法。

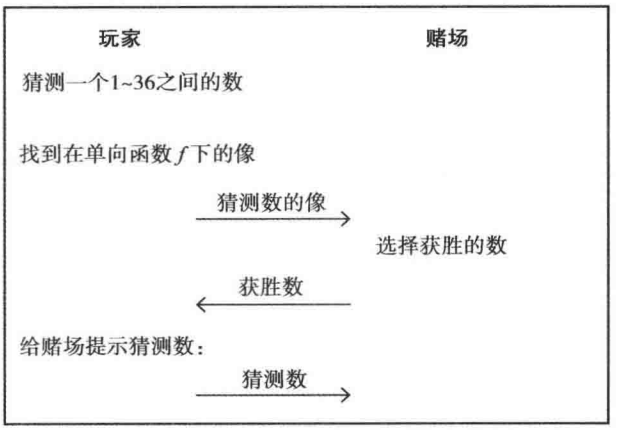
提示：当窃听到A、B与可信中心之间的通信之后，C请求与D通信。

6. 考虑一个因特网轮盘游戏。一个玩家下注，选择1到36之间的一个数，然后将选择的数发送给网上赌场。赌场选择一个数（可能是随机的!），然后告诉玩家是否赢了。很显然，如果玩家将他选择的数以明文形式发送过去，则会存在一个安全问题：赌场可以直接不选择这个数!

这里有一个显然更好的实现轮盘游戏的方式。有一个全系统的单向函数f。玩家将他的猜测作为f的输入，计算相应的输出并发送给赌场，而不是直接发送他猜测的数。一旦赌场将获奖数发送给玩家，玩家便告诉赌场他的猜测。赌场可以验证玩家没有说谎，因为他可以得到玩家声称的猜测在f下的像，然后验证和玩家之前发送的数是否一致。

(a) 描述这个系统的最大安全缺陷。

(b) 假设f是一个定义域在1到10^{20}之间的可逆函数。提出一种修复方法使得该系统安全。



延伸阅读

[1] E. T. Bell. *Men of Mathematics*. New York, NY: Simon and Schuster (1937).

[2] David Kahn. *The Codebreakers: The Story of Secret Writing*, 2nd ed. New York, NY: Scribner (1996).

[3] Peter Wright with Paul Greengrass. *Spycatcher: The Candid Autobiography of a Senior Intelligence Officer*. New York, NY: Viking (1987).

[4] Charlie Kaufman, Radia Perlman, and Mike Speciner. *Network Security: Private Communication in a Public World*. Englewood Cliffs, NJ: Prentice—Hall (1995).

[5] Bruce Schneier. *Applied Cryptography*, 2nd ed. New York, NY: John Wiley & Sons (1996).

[6] Douglas R. Stinson. *Cryptography: Theory and Practice*. Boca Raton, FL: CRC Press (1995).

索　引

索引中的页码为英文原书页码，与页边标注的页码一致。